Physical Database Design and Database Tuning

Bright House Apartment

Project Phase 4

By

Mr. Anon	Kangpanich	6088053
Ms. Tanaporn	Rojanaridpiched	6088146
Mr. Tanawin	Wichit	6088221
Mr. Pornthep	Duangdarw	6088038

A Report Submitted in Partial Fulfillment of

the Requirements for

ITCS413 - Database Design

Faculty of Information and Communication Technology

Mahidol University

Semester 2/2019

Table of Contents

Data Requirements & Required Transactions

In this section, the user's data requirements including a description of required transactions

> ORIGINAL DELIVERABLE REQUIREMENT
>
> a. The data requirements including description of required transactions

from the first phase of the project will be listed.

Data Requirements

Please note that the data requirements listed below are the combined summary of `BranchManagement`, `BranchEmployee`, and `Customer` User's views; therefore, requirements on duplicate or similar entity or data will be combined or simplified into one.

Branch

A **branch** refers to an operational unit of Bright House. There are 10 branches of the Bright House apartment scattered throughout the eastern region of Thailand. Each branch has multiple employees and managed by the so-called manager or a branch manager. Each branch has a unique name, up to 3 **telephone** or fax **numbers**, a location address that includes street name, subdistrict, district, and province. Lastly, each branch also has an identification number as well as a unique contact email address that can facilitate communication with customers.

Employee

An **employee** refers to a workforce that operates, drives, and facilitate each branch in its operations. An employee cannot operate at multiple branches unless there is a request from their manager. Currently, there are 4 possible roles: **Manager** (sometimes called Branch Manager or Apartment Manager), **Cleaning Personnel**, **Accountant**, and **Security**. Each branch can also have a manager and an accountant. Each role determines daily routine or duties during the **working hour** of an employee. Each employee will be given a unique identification number. Each employee must also provide their personal information that includes full name, nickname, date of birth, gender, home address, daily wage, citizen ID, nationality, email address, and up to 3 **phone numbers**.

Customer

A **customer** refers to the one who may reserve, rent, and stay at a room in one of Bright House's branches. When a customer rents a room, he or she can stay as long as the period is specified in the rental detail. During rent, a customer may request **additional services** at an additional expense such as item refilling or cleaning. When a customer visits one of a branch for the first time, he or she will be asked to fill information that consists of: first name, middle name, last name, nickname (if any), date of birth, gender, homeland country, city, profession, citizen ID (if any), nationality, email address, **up to 3 phone numbers**, as well as **passport** details, and **visa** details. As for customers who already registered, they will be asked for their passport details and visa details only. Lastly, each customer can also provide employees **feedback** each of which will be reviewed by a manager.

Supply Inventory

Supply Inventory refers to an **inventory** that each branch owns. An inventory can store items such as **consumable** and **non-consumable items or supply**. This also includes furniture or any physical objects that each branch stockpiles to help in its operation. Each transfer or **transaction** will be recorded in a branch-owned **log** and can be either be inbound or outbound. Additionally, for each item transfer audit record, the information about **responsible and supervising persons** will also be required. Each entry may also include a timestamp and transferred quantity. Please note that each entry needs to get recorded and **reviewed by the accountant** of each branch.

Room Service

Room service refers to a service that can be done by either a customer **request** or a **routine** that is **scheduled by an interval**. There are 2 types of room services: cleaning and resupplying. Cleaning service has many **actions/tasks** all of which will be **done by cleaning personnel**. The **actions** may include table wiping, floor-sweeping, floor mopping, bathroom cleaning, towel changing, sheets, and pillowcases changing. Each cleaning personnel is also required to keep track of the tasks done during room service. Lastly, a customer in a rental may request an **additional room service** with an additional charge. Additionally, each room service may also **involve multiple instances of inventory log** or entries in the case that there is additional equipment or consumable that is required for the service.

Room

A **Room** refers to a unit of accommodation provided for customers to temporary rent. Each room differs in terms of size, furniture, and location; therefore, each room needs a room

number that can uniquely identify each room in a branch. Each room can only be presented at a branch. Please note that the size of a room is a factor that affects the rental fee.

Customer Payment

A **customer payment**, also known as a customer billing, refers to an instance of payment, that **includes multiple items** or **lines**, that a customer of a rental has to pay to Bright House. **An item in payment may include** but not limited to the rental fee, room utility charges, service charges, property damage penalty fee. Accountants and Managers in each branch can use this information to gain insight into its creation time, amount, description, and the associated transaction of each entry.

Room Utility

A **room utility** refers to a utility such as water or electricity that is bound to a room. Typically, an employee has to record utility meters to subsequently determine the electricity and tap water usage of a rental tied to the room. The usage of water and electricity in the unit will be later **calculated by using a standard rate**. The result is the fee that the customer in that rental has to pay. To determine the usage, an employee must subtract two instances of meter readings from two different points of time; therefore, recording time, room number, rental number, electricity and water meter readings have to be kept.

Reservation

A **reservation** refers to a reservation inquiry or proposal message sent from a customer to a branch via an email or other contacts. Each user **can reserve more than one room** as long as they are available during the desired time. The details of a reservation include a uniquely identifiable number, the details of the person who make the reservation, the room numbers, remarks as well as the details of the staff who inquires the reservation.

Rental

A **rental** refers to a continuous period of time that consists of multiple sub-periods each of which can be either a day- or month-long period depending on **the rental type** (e.g. daily or monthly). A rental can be, but not necessarily be, a consequence of a reservation. Each rental period **can also have multiple bills** depending on the circumstances. Please note that rental is tied to only a room; thus, if a customer wants to move to another room, a new rental assign to the new room has to be created. Rental information is also useful when an employee wants to search for a free room. A rental may consist of **up to 2 customers.**

Payroll

A **payroll**, in this case, refers to **wage payment** for employees. Since each employee can receive a wage at a different rate, all wage payments need to be recorded by the manager of a branch. Each wage payment will be calculated based on the actual daily working hours in each week. An individualized wage payment rate will multiply the total working hour of each week to get the final amount of money. Each employee can query to get a report of their payrolls to ensure the transparency of the payroll.

Financial Transaction

A **financial transaction** refers to a monetary transaction related to the business. This can include transactions caused by a wage payment, a customer payment, or any other spending made by a branch such as purchasing. An accountant in each branch is responsible to review each financial transaction to ensure integrity. Details such as description, transaction direction (e.g. inbound or outbound), amount, record created time, transaction time, record last modified time, status, and category (e.g. Deposit or Return) will be recorded for each transaction.

Maintenance

Maintenance refers to a task carried out by an external mechanic or an employee to repair an object or a property of a branch that is **damaged**. Any employee can request for maintenance for any object. Moreover, recording maintenance task helps branch managers and employees to keep track of tasks that sometimes take an extensive amount of time to finish. Each maintenance will be recorded in terms of record creation time, maintenance starting time, finished time, last modified time, description, category, and the requestor. Please also note that if property damage is done by a customer and the damage is severe, the customer will be charged for the damage.

Purchasing

In order for a branch to operate normally, it relies on externals products such as washing powder, floor wax, etc.; as a consequence, **purchasing** becomes a regular task. Each instance of purchasing will be recorded in terms of **vendor**, status, type, created time, approval time, rejected time. The level of details of each purchasing will be each **purchasing line** which **includes only one product or supply**. Before any purchase can take place, it must be approved by either a branch manager or an accountant. Multiple employees can participate in purchasing if asked by a manager.

Required Transactions

1. BranchManagement User's View

1.1. DATA ENTRY

1.1.1. Enter the details of a new **Branch** (such as branch ID 1; "Tanachon", manager ID 1, ...).

1.1.2. Enter the details of a new Member of an **Employee** at a branch (such as Supachai Jaidee).

1.1.3. Enter the details of a new **Customer** who check-in for the first time at the front counter at a branch. (such as Peter Tarlay).

1.1.4. Enter the details of a new **Room** in a Branch (such as room number 203 is a superior room, which consists of a bed, a television, a wardrobe, and a safe, located on the 2nd floor at Tanachon branch).

1.1.5. Enter the details of new **Customer Payment** (such as payment for room number 203 rented by Peter Tarley).

1.1.6. Enter the details of a new **Room Utility** record (such as an electricity utility record of room number 203 on 23/5/2019 is 3486 units).

1.1.7. Enter the details of a new **Reservation** record received from a customer email or phone call (such as a reservation record for Peter Tarlay booked on 10/5/2019).

1.1.8. Enter the details of a new **Rental** at a branch (such as customer Peter Tarlay is renting and staying at room number 203 at Tanachon branch as a monthly rental started 23/05/2019).

1.1.9. Enter the details of a new employee work attendance for the calculation of **Payroll** at a branch (For example, Mr. Supachai Jaidee comes to work at the Tanachon branch at 07:00, and leaves the work at 21:00).

1.1.10. Enter the details of a new **Maintenance** instance that occurred within a branch in terms of maintenance description, requestor name, and position. Furthermore, start and ending time, operatives who carry out the maintenance and supply used will also be recorded.

1.1.11. Enter the details of a new **Purchasing** requested by anyone who is a branch employee. Requestor information, a brief description, a person-in-charge list, **vendor** info, and item list (only item description and unit price are required) must be specified.

1.2. DATA UPDATE/DELETION

1.2.1. Update/delete the details of a **Branch**.

1.2.2. Update/delete the details of a member of **Employee** at a branch.

1.2.3. Update/delete the details of a **Customer** including feedback.

1.2.4. Update/delete the details of a **Room** in a Branch.

1.2.5. Update/delete the details of a given **Customer Payment** at a given branch.

1.2.6. Update/delete the details of a **Room Utility** at a given branch.

1.2.7. Update/delete the details of a room **Reservation** made by a customer at a given branch.

1.2.8. Update/delete the details of a room **Rental** at a given branch.

1.2.9. Update/delete the details of a **Payroll** of a given employee at a given branch.

1.2.10. Update/delete the details of a **Maintenance** at a given branch. An update will occur when there is a change in maintenance status.

1.2.11. Update/delete the details of a **Purchasing** at a given branch. An update of a purchasing will happen after the person-in-charge has proceeded to purchase the items. An update will include time purchased, discounts, and unit price (if any).

1.3. DATA QUERIES

1.3.1. List the details of **branches** in a given area of the Eastern region of Thailand.

1.3.2. List all details of **employees** given an employee ID, name, surname, or branch name.

1.3.3. Identify the total number of **employees** in all branches.

1.3.4. Identify the average salary of **employees** in each position, gender, branch, ordered in ascending order.

1.3.5. List all active branch **managers** ordered by the branch address.

1.3.6. List **employee** details by a given position, name, ID, and Date of birth.

1.3.7. Identify the total **employee** payroll for a day, week, or month.

1.3.8. Identify the total number of unique **customers** that have ever visited at least one branch.

1.3.9. List unique **customers** that have ever visited at least one branch.

1.3.10. Identify the average age of **customers** from each nationality.

1.3.11. List **customer** details grouped by a given branch.

1.3.12. List **customer** details by a given duration from their last visited time ordered by the duration.

1.3.13. List details of **customers** who recently stay more than one branch of Bright House ordered by descending order.
1.3.14. List details of **supplies** that are used the most by any branches.
1.3.15. Identify the average number of **supplies** left at the end of every week grouped by each the name of the supply.
1.3.16. List room details with a status of the **rental** by a given branch or area.
1.3.17. List all **rooms** that currently have guests staying grouped by room size.
1.3.18. List the details of customer **payment transactions** along with full customer name and nationality that has the top-10-most transaction amounts in a given month.
1.3.19. Identify the average **customer payment** amount grouped by transaction type.
1.3.20. Identify the average amount of money in **customer payments** grouped by customer's nationality, and ordered by month.
1.3.21. Identify the **customer payment** that has the highest amount of money grouped by month, customer's nationality.
1.3.22. List the **customer payment** by a given rental number and customer name.
1.3.23. Identify the average amount of **customer payments** grouped by month.
1.3.24. Identify the frequency of **rental** grouped by month.
1.3.25. List all **rentals** that will be **expired** within a given time range at a branch.
1.3.26. List all **rooms** that will be **free** within a given time range at any branch in a case the customer must be redirected to another branch.
1.3.27. List all **rentals** in any branches associated with a given **customer**.
1.3.28. List all **room services** done to a specified **room** within a branch.
1.3.29. List all **room services** done to a specified **rental** within a branch.
1.3.30. Identify the average duration of **customer rentals** within any branch.
1.3.31. Identify the average duration of stay of all **customer** Visa within any branch.
1.3.32. List all **branches** ordered by the total number of free rooms within a given time range.
1.3.33. Identify **months** that have a high amount of reservations.
1.3.34. Identify **months** that have a high amount of income.
1.3.35. Identify the difference between income from rentals & additional cleaning services and expenses from room supply during high and low seasons.
1.3.36. List **feedback** reported by **customers** categorized by branches.
1.3.37. Identify the total number of customer **feedback** in each branch.
1.3.38. Identify the total number of customer **feedback** grouped by categories.

1.3.39. Identify the average number of customer **feedback** per rental.
1.3.40. List customer **feedback** from all branches in a given period of time.
1.3.41. List all **payrolls** made within a given period of time.
1.3.42. List all **employees** that have total **payroll** within a given threshold range.
1.3.43. Identify the average employee **payrolls** in a given time range.
1.3.44. List all **maintenance** instances of a branch based on a given maintenance status.
1.3.45. For each branch, list all supplies used in all **maintenance** so far.
1.3.46. List all **external operatives** involved in any **maintenance** instance.
1.3.47. List all completed **maintenance** instances that took more than a given time duration to complete.
1.3.48. List all ongoing **maintenance** instances that take more than a given time duration so far.
1.3.49. List all on-going **purchasing** instance and its items.
1.3.50. Identify the average of a given item out of all **purchasing** instances that contain it.
1.3.51. Identify the total sum of money used in **purchasing** instances within a given time range.
1.3.52. List all **items** that are included in any **purchasing** instances in a given time range.

2. BranchEmployee User's View

2.1. DATA ENTRY

2.1.1. Enter the data of a **Room Service** (such as room 203 with the superior room size and being rent monthly will receive "น้ำเปล่าขวดใหญ่", "กาแฟซอง", "สบู่ก้อน", "หมวกคลุมผม". Each with a quantity of 2).
2.1.2. Enter the details of a new transaction of **Supply** (such as "น้ำเปล่าขวดใหญ่ แพ็ค" is being transferred into Tanachon Branch with the quantity of 10 pcs. by Supachai Jaidee).
2.1.3. For accountants, Enter the details of a new **Financial Transaction** made from either customer rental transactions, payroll transactions, property damage fine, or purchasing.
2.1.4. For accountants, Enter the details of a new **Purchasing** requested any employees likewise to what a branch manager can.

2.2. DATA UPDATE/DELETION

2.2.1. Update/delete the details of a **Room Utility** at a given branch.
2.2.2. Update/delete the details of a **Customer Payment** at a given branch.
2.2.3. Update/delete the details of a **Room** at a given branch.
2.2.4. Update/delete the details of a **Supply** at a given branch.
2.2.5. Update/delete the details of a **Room Service** at a given branch.
2.2.6. Update/delete the details of a **Financial Transaction** specific to a branch. An update of a purchasing will happen after an accountant reviewed a transaction and flagged it as reviewed.
2.2.7. Update/delete the details of a **Purchasing** at a given branch. An update of a purchasing will happen after the person-in-charge has proceeded to purchase the items. An update will include time purchased, discounts, and unit price (if any).

2.3. DATA QUERIES

2.3.1. List the number, full name, working hours, position of **Employees** by a given branch sorted by name alphabetically.
2.3.2. List the details of **Supplies** by a given branch and supply name, ordered by quantity in ascending order.
2.3.3. List details of **supplies** that are used the most by a branch.
2.3.4. List the details of **Room Services** for cleaning personnel.
2.3.5. List the number, floor, and building of **Rooms** that have occupied by customers.
2.3.6. Identify the total sum of **Customer Payment** within a given period in a branch.
2.3.7. Identify the **Room Utility** cost for a monthly rental within a branch.
2.3.8. Identify the total **Room Utility** cost for all monthly rentals within a branch.
2.3.9. List all **Utility** records of a given Room and given rental number.
2.3.10. List the room number, check-in time, check-out time, and type of **Rental** for the process of planning and scheduling room services, and security routine.
2.3.11. List **Customer Payments,** that associates with a branch, by time and transaction type.
2.3.12. List room number, customer name, rental type, deposit, rental amount, as well as **supplies** used by the **room**.
2.3.13. List all **rentals** that will be **expired** within a given time range at a branch.

2.3.14. List all **rooms** that will be **free** within a given time range at any branch in a case the customer must be redirected to another branch.
2.3.15. Identify room number and rental type of branch **reservations** that are scheduled within a given time range.
2.3.16. List all **room services** done to a specified **room** within a branch.
2.3.17. List all **room services** done to a specified **rental** within a branch.
2.3.18. List all **room services** done by a given **employee** in a branch.
2.3.19. List all **rental check-in** that is supervised by a given **employee**.
2.3.20. List all **rental check-out** that is supervised by a given **employee**.
2.3.21. List periods that have the maximum number of cleaning service requests within a branch ordered by descending order.
2.3.22. Identify the average number of unique **customers** per week within a branch in a given time range.
2.3.23. Identify the average number of new **customers** per week within a branch in a given time range.
2.3.24. Identify which **room** has the longest rental duration from any customer with a given period.
2.3.25. Identify which periods will have the highest **utility** usages in terms of unit or cost in a branch.
2.3.26. List items to be purchased in a given on-going **purchasing** instance.
2.3.27. List **maintenance** instances within a given time range.
2.3.28. List **financial transactions** made in a branch with a given period of time.
2.3.29. List **payrolls** that are associated with a given employee.
2.3.30. Identify the total sum of money in all payrolls that associated with a given **employee**.
2.3.31. List all **Maintenances** and preceding inspections.

3. Customer User's View

3.1. DATA ENTRY

Customers do not enter the data by themselves. It is the duty of a branch manager.

3.2. DATA UPDATE/DELETION

Customers do not update/delete the data by themselves. It is the duty of a branch manager.

3.3. DATA QUERIES

3.3.1. List the details of **Rooms** in each branch.

3.3.2. List the details of a **Reservation** that a particular customer has reserved. (A customer can reserve a room and get the details of the Reservation that they have made)

3.3.3. List the details of **Branches** by a location given by a customer.

3.3.4. List the details of **Available rooms** in each branch.

3.3.5. List the **Contact** details in each branch.

3.3.6. List the **Feedback** that is submitted by the customer who gave the feedback.

Implemented SQL Stored Procedures, Triggers & Views

NOTE

This optional section is written to help readers in navigating, searching or understanding of SQL commands that are created as part of this project.

Naming Convention

Before started implementing various SQL objects, our team agree to create a naming convention specifically used in this project to enable consistency. In this sub-section, the naming convention will be explained.

Function (Abbreviated as F)

Uses Pascal case naming with **"fun" prefix** to indicate that the object is **a function**. For example, ***`funFormatFullNameString`***.

Stored Procedure (Abbreviated as P)

Uses Pascal case naming with **"sp" prefix** to indicate that the object is **a stored procedure**. In this project, there are 2 types of procedure:

1. **Modification Procedures** – The ones that manipulate data by either Insert / Update / Delete. These procedures use the format:

```
sp + <Verb> + <Subject>
```

For example, ***spInsertBranch***, ***spInsertSubsequentMonthlyRentalPeriod***.

2. **Data Retrieval Procedures** – The ones that retrieve data from a given set of parameters. These procedures use the format:

   ```
   sp + <List/Find/Get/...> + <Subject> + By + <Condition Subject>
   ```

   For example, ***spListBranchByRegion***.

Trigger (Abbreviated as T)

Uses Pascal case naming with **"tr" prefix** to indicate that the object is **a trigger**.

```
tr + <Add/Update/Delete> + <Subject>
```

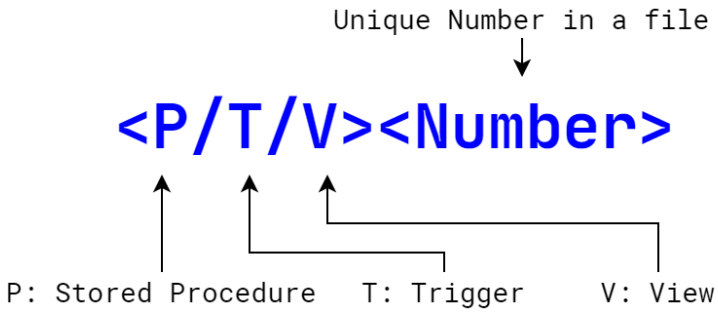
For example, ***trAddRoomUtilityOperand***, ***trAddRentalPeriod***.

View (Abbreviated as V)

Uses Pascal case naming with **"View" suffix**. For example, ***BranchAvgRentalAndCustomerView***.

Numbering

Our team organized SQL script into 3 files classified by their type: Stored Procedure, Trigger, and View. Each command in a file is assigned a number that is unique within the file; therefore, it means that there will be number duplicates across 3 files. To identify them globally across the file, we will use the following format:



For example, to identify the stored procedure named ***spInsertRental*** which is the **8th** procedure in the procedure SQL file, we can identify it as "**P8**".

Implementation List

Since the way our team implements each SQL object (e.g. procedure, trigger or view) is based on multiple user transaction requirements instead of one user transaction requirement per one SQL object, in this section, the mapping between each SQL object to many transaction requirements will be listed in a tabular manner. Please also note that not all requirements are implemented due to a limited time constraint.

NOTE

Readers can use this section to identify how each user requirement is related to each SQL object.

***Related User Requirements**

This column shows the mapping between a SQL implementation to user requirements listed in the previous section. For example, **(1.1.1) refers to the requirement that states,** ***"Enter the details of a new Branch (such as branch ID 1; "Tanachon", manager ID 1, ...)."***

ID	Name	Description	Related User Requirements *
P1	spInsertNewBranch	Insert the details of a new Branch	(1.1.1)
P2	spInsertEmployee	Insert the details of a new Member of an Employee	(1.1.2)
P2.1	spInsertWorkingShift	Insert the employee working shift detail	(1.1.2)
P3	spInsertCustomer	Insert the details of a New Customer	(1.1.3)
P3.1	spInsertCustomerPassport	Insert the details of customers Passport	(1.1.3)
P3.2	spInsertCustomerVisa	Insert the details of customers Visa	(1.1.3)
P4	spInsertNewRoom	Insert the details of a new Room in a Branch	(1.1.4)
P5	spInsertCustomerPaymentBillingLine	Insert the details of a new Customer Payment Billing line	(1.1.5)

ID	Name	Description	Related User Requirements *
P6	spInsertUtilityCost	Insert the details of a new Room Utility	(1.1.6)
P7	spInsertReservation	Enter the details of a new Reservation	(1.1.7)
P8	spInsertRental	Enter the details of a new Rental	(1.1.8)
P9	spInsertSubsequentMonthlyRentalPeriod	Enter Subsequent Monthly Rental Period	(1.1.8)
P10	spInsertCustomerRequestedRoomService	Enter the data of a Requested Room Service	(2.1.1)
P10.1	spGetRentalDuringTimeByCustomerNum	Get Current Rental by a Customer	
P10.2	spGetLatestCustomerBillingByRentalNum	Get Current CustomerBill by RentalNum	
P10.3	spGetLatestBillingLineByBillingNum	Get Latest BillingLine by BillingNumber	
P11	spInsertActionDoneInRequestedRoomService	Enter the data of an Action done within a session of Requested Room Service	(2.1.1)
P12	spInsertWithdrawalEntryRelatedToCustomerService	Add a Supply Withdrawal entry associated with Customer Service.	
P12.1	spIdentifyReturningSupplyOnHandQuantity	Retrieve the number of a returning supply so far given a branch.	(2.3.2)
P12.2	spFindBranchOfGivenEmployeeNum	Find Branch Number from a given employee number	
P12.3	spFindAccountantByBranchNum	Find accountant number by a branch number	
P13	spInsertRoutineRoomServiceAction	Enter the data of an Action done with a Routine Room Service	(2.1.1)
P14	spInsertNewSupplyFromPurchasingToInventory	Add a new Supply from a purchasing to the inventory	(2.1.2)

ID	Name	Description	Related User Requirements *
P15	spListBranchByRegion	List branches by a region	(1.3.1), (1.3.3), (1.3.5), (3.3.3)
P16	spListEmployeeByGivenDetails	List employee by given details	(1.3.2), (1.3.6)
P17	spListCustomerRentalsByTimeWindow	For each Customer, list their rentals that are within the given time window.	(1.3.13)
P18	spIdentifyCustomersByRentalDetails	Identify Customers by Rental Details	(1.3.22)
P19	spListExpiringRentalsInTimeRange	List expiring rentals in time range	(1.3.25)
P20	spListFreeRoomInGivenTimeRange	List free rooms in a given time range	(1.3.26)
P21	spListAllRentalsByCustomerNumber	List all rentals in any branches associated with a given customer	(1.3.27)
P22	spListRoomServiceByRental	For a given rental, list all room service details including its category, total actions are done, starting time.	(1.3.29), (2.3.17)
P23	spInsertPropertyInspection	Insert the details of a Property Inspection	(1.1.10)
P24	spInsertPropertyDamage	Insert the details of a Property Damage	(1.1.10)
P25	spInsertMaintenanceTask	Insert the details of a Maintenance	(1.1.10)
P26	spGetBranchEmployeeWithMaxWage	Get Branch employee with a max wage	
P27	spInsertRoutineRoomService	Enter the data of a Routine Room Service	(2.1.1)
P28	spListCustomerPaymentByRentalCustomer	List the customer payment by a given rental number and customer name.	(1.3.22)
T1	trAddRoomUtilityOperand	Room Utility Operand Trigger	(1.1.6)
T2	trAddRentalPeriod	Initial Rental Period Trigger	(1.1.8)

ID	Name	Description	Related User Requirements *
V1	EmployeeDetailsView	List the details of each employee including gender, branch, wage, position, working hours.	(1.3.4), (2.3.1)
V2	EmployeePayrollView	Identify the total employee payroll for a day, week, or month.	(1.3.7)
V3	CustomerView	Identify customer information as well as his/her first rental branch.	(1.3.8), (1.3.9) (1.3.11), (1.3.12)
V4	SummarySupplyView	List the details of each type of supply for each branch	(1.3.14)
V5	WeeklySupplySummaryView	Show the summary of supply weekly usage in each branch	(1.3.14), (1.3.15)
V6	RoomRentalStatusView	List every room in all branches as well as identify its occupation or reservation status	(1.3.16), (1.3.17), (3.3.1), (3.3.4)
V7	AvgCustomerDemographicPaymentView	List Average Payment is done by customers in each demographic (or nationality).	(1.3.10), (1.3.20)
V8	MonthlyHighestCustomerPaymentView	List Highest Customer Payment grouped by Month and Customer's nationality	(1.3.21)
V9	MonthlyAvgCustomerPaymentView	List Average Customer Payment grouped by Month and Customer's nationality	(1.3.23)
V10	MonthlyRentalFrequencyView	List the frequency of starting and the frequency ending rentals in each month	(1.3.24)
V11	IndividualRoomServiceView	For each service instance, list all room service details including its category, total actions done, starting time.	(1.3.28)
V12	BranchAvgRentalAndCustomerView	For each branch, retrieve information regarding rentals and	(1.3.30)

ID	Name	Description	Related User Requirements *
		customers such as rental duration, PAX, etc. Then summarize them with average or total sum.	
V13	MaintenancePrecedingInspectionView	List all Maintenance instances and preceding inspections	(2.3.31)
V14	TopTenMonthlyBranchCustomerTransactionView	List Top-10 customer transactions in each month along with associated customer details	(1.3.18)
V15	RentalSupplyView	List room number, customer name, rental type, deposit, rental amount, as well as supplies used by the room.	(2.3.12)
V16	CustomerReservationView	List the details of a Reservation that customers have made.	(3.3.2)
V17	BranchContactView	List the Contact details in each branch	(3.3.5)
V18	CustomerFeedbackView	List the Feedback that is submitted by the customer who gave the feedback	(3.3.6)

Table 1 – List of all transactions implemented along with related user requirements that are specified the Required Transaction section.

Additional Note

Most of the implemented **View**s are implemented in a way that it contains many aggregated columns; therefore, they cannot be updated or modified.

Furthermore, some of the transactions may not be associated with any required user requirement since they are created as a helper for other procedures.

BRIGHT HOUSE
CONCEPTUAL MODEL UML ERD
Finalized on 25 Feb 2020

The final ER Diagram

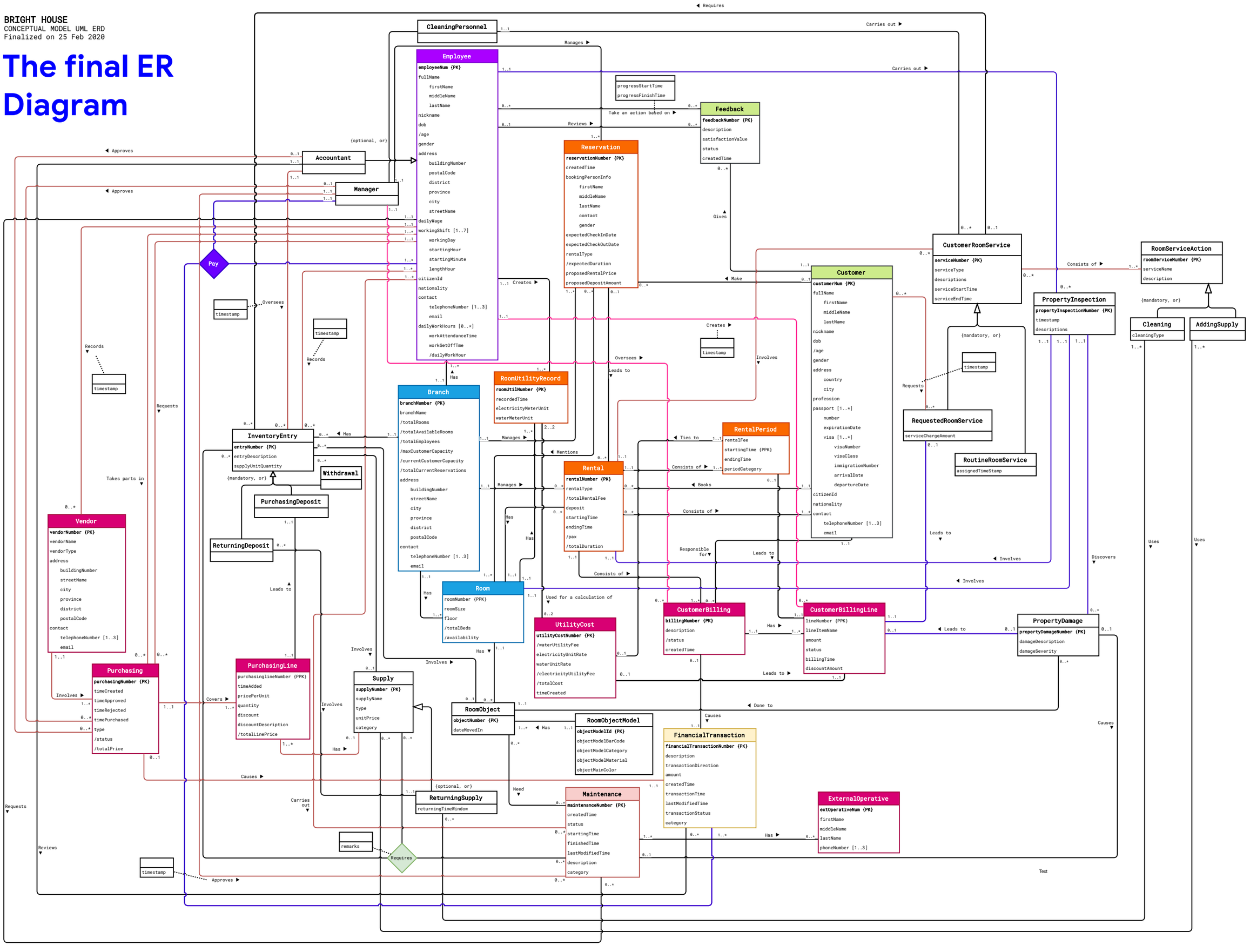


Figure 1 – The updated version of the ER Diagram of the Bright House database

The final Relational database schema

Supply(supplyNumber, supplyName, supplyType, supplyUnitPrice, supplyCategory) **Primary Key:** supplyNumber	**ReturningSupply**(supplyReturningMinuteTimeWindow, returningSupplyNumber) **Primary Key:** returningSupplyNumber **Foreign Key:** returningSupplyNumber
Customer(customerNum, customerFirstName, customerMiddleName, customerLastName, customerNickname, customerDOB, customerGender, customerCountry, customerCity, customerProfession, customerCitizenId, customerNationality, customerEmail) **Primary Key:** customerNum	**Branch**(branchNumber, branchName, branchBuildingNumber, branchStreetName, branchSubdistrict, branchDistrict, branchProvince, branchPostalCode, branchEmail) **Primary Key:** branchNumber
BranchTelephone(branchTelephoneNumber, telBranchNumber) **Primary Key:** branchTelephoneNumber **Foreign Key:** telephoneBranchNumber	**Room**(roomBranchNumber,roomNumber ,roomSize,roomFloor) **Primary Key:** (roomBranchNumber,roomNumber) **Foreign Key:** roomBranchNumber
Employee(employeeNum, employeeFirstName, employeeMiddleName, employeeLastName, employeeNickname, employeeDOB, employeeGender, employeeBuildingNumber, employeeStreetName, employeeSubdistrict, employeeDistrict, employeeProvince, employeePostalcode, dailyWage, employeeCitizenId, employeeNationality, employeeEmail, employeeBranchNumber) **Primary Key:** employeeNum **Alternate Key:** employeeCitizenId **Foreign Key:** employeeBranchNumber	EmployeeWorkingShift(workingDay, workingShiftStartingHour, workingShiftStartingMinute, workingShiftLengthHour, workingShiftEmployeeNum) **Primary Key:** (workingDay, workingShiftStartingHour, workingShiftStartingMinute, workingShiftLengthHour, workingShiftEmployeeNum) **Foreign Key:** workingShiftEmployeeNum
EmployeeTelephone(employeeTelephoneNumber, telephoneEmployeeNum) **Primary Key:** employeeTelephoneNumber **Foreign Key:** TelephoneEmployeeNum	EmployeeDailyWorkHours(workAttendanceTime, workGetOffTime, workHourEmployeeNum) **Primary Key:** (workAttendanceTime, workGetOffTime, workHourEmployeeNum) **Foreign Key:** workHourEmployeeNum
CleaningPersonnel(cleaningPersonnelNum) **Primary Key:** cleaningPersonnelNum **Foreign Key:** cleaningPersonnelNum	**Manager**(ManagerNum) **Primary Key:** managerNum **Foreign Key:** managerNum

Accountant(AccountantNum) **Primary Key:** accountantNum **Foreign Key:** accountantNum	**FinancialTransaction**(financialTransactionNumber, financialTransactionDescription, financialTransactionDirection, financialTransactionAmount, financialTransactionCreatedTime, financialTransactionTime, financialTransactionLastModifiedTime, financialTransactionStatus, financialTransactionCategory, financialTransactionAccountNum) **Primary Key:** financialTransactionNumber **Foreign Key:** financialTransactionAccountNum
Maintenance(maintenanceNumber, maintenanceCreatedTime, maintenanceStatus, maintenanceStartingTime, maintenanceFinishedTime, maintenanceLastModifiedTime, maintenanceDescription, maintenanceCategory, maintenanceRequestorNum) **Primary Key:** maintenanceNumber **Foreign Key:** maintenanceRequestorNum	**Reservation**(reservationNumber, reservationCreatedTime, bookingFirstName, bookingMiddleName, bookingLastName, bookingContact, bookingGender, expectedCheckInDate, expectedCheckOutDate, rentalType, proposedRentalPrice, proposedDepositAmount, reserveBranchNumber, reserveCustomerNumber, reserveManagerNumber) **Primary Key:** reservationNumber **Foreign Key:** reserveBranchNumber **Foreign Key:** reserveCustomerNumber **Foreign Key:** reserveManagerNumber
Rental(rentalNumber, rentalType, rentalDeposit, rentalStartingTime, rentalEndingTime, rentalBranchNumber, rentalRoomNumber, rentalBookingCustomerNum, rentalReservationNumber) **Primary Key:** rentalNumber **Foreign Key:** rentalBookingCustomerNum **Foreign Key:** rentalReservationNumber **Foreign Key:** (rentalBranchNumber, rentalRoomNumber)	**CustomerBilling**(billingNumber, billingDescription, billingCreatedTime, billingManagerNum, billingRentalNumber, billingCustomerNum, billingFinancialTransactionNumber) **Primary Key:** billingNumber **Foreign Key:** billingManagerNum **Foreign Key:** billingRentalNumber **Foreign Key:** billingCustomerNum **Foreign Key:** billingFinancialTransactionNumber
CustomerBillingLine(customerBillingNumber, customerBillingLineNumber, customerBillingLineItemName, customerBillingAmount, customerBillingStatus, customerBillingTime, customerDiscountAmount, customerBillingEmployeeId, customerBillingCreatedTimestamp) **Primary Key:** (customerBillingNumber, customerBillingLineNumber) **Foreign Key:** customerBillingNumber **Foreign Key:** customerBillingEmployeeId	**Feedback**(feedbackNumber, feedbackDescription, satisfactionValue, feedbackStatus, feedbackCreatedTime, feedbackEmployeeNum, feedbackCustomerNum) **Primary Key:** feedbackNumber **Foreign Key:** feedbackEmployeeNum **Foreign Key:** feedbackCustomerNum
FeedbackEmployeeAction(actionFeedbackNumber, actionEmployeeNum, progressStartTime, progressFinishTime) **Primary Key:** (actionFeedbackNumber, actionEmployeeNum) **Foreign Key:** actionFeedbackNumber **Foreign Key:** actionEmployeeNum	**RoomUtilityRecord**(roomUtilNumber, recordedTime, electricityMeterUnit, waterMeterUnit, roomUtilEmployeeNum, roomUtilBranchNumber, roomUtilRoomNumber) **Primary Key:** roomUtilNumber **Foreign Key:** roomUtilEmployeeNum **Foreign Key:** (roomUtilBranchNumber, roomUtilRoomNumber)

RentalCustomer(rentalCustomerRentalNumber, rentalCustomerNum) **Primary Key:** (rentalCustomerRentalNumber, rentalCustomerNum) **Foreign Key:** rentalCustomerRentalNumber **Foreign Key:** rentalCustomerNum	**RentalPeriod**(rentalPeriodRentalNumber, rentalPeriodStartingTime, rentalPeriodEndingTime, periodCategory, rentalFee, rentalPeriodBillingNumber, rentalPeriodLineNumber) **Primary Key:** (rentalPeriodRentalNumber, rentalPeriodStartingTime) **Foreign Key:** rentalPeriodRentalNumber **Foreign Key:** (rentalPeriodBillingNumber, rentalPeriodLineNumber)
UtilityCost(utilityCostNumber, electricityUnitRate, waterUnitRate, utilityTimeCreated, utilityRentalNumber, utilityStartingTime, utilityBillingNumber, utilityLineNumber) **Primary Key:** utilityCostNumber **Foreign Key:** (utilityBillingNumber, utilityLineNumber) **Foreign Key:** (utilityRentalNumber, utilityStartingTime)	**UtilityCostOperand**(operandUtilityCostNumber, operandRoomUtilNumber) **Primary Key:** (operandUtilityCostNumber, operandRoomUtilNumber) **Foreign Key:** operandUtilityCostNumber **Foreign Key:** operandRoomUtilNumber
CustomerTelephone(customerTelephoneNumber,telephoneCustomerNum) **Primary Key:** customerTelephoneNumber **Foreign Key:** telephoneCustomerNum	**CustomerPassport**(passportNumber, expirationDate, passportCustomerNum) **Primary Key:** passportNumber **Foreign Key:** passportCustomerNum
CustomerVisa(visaNumber, visaClass, immigrationNumber, arrivalDate, departureDate, visaPassportNumber) Primary Key: visaNumber **Alternate Key:** immigrationNumber **Foreign Key:** visaPassportNumber	**CustomerRoomService**(serviceNumber, serviceType, serviceDescription, serviceStartTime, serviceEndTime, serviceRentalNumber, serviceCleaningPersonalNum) **Primary Key:** serviceNumber **Foreign Key:** serviceRentalNumber **Foreign Key:** serviceCleaningPersonalNum
RequestedRoomService(requestedBillingNumber, requestedLineNumber, requestedServiceChargeAmount, requestedServiceNumber) **Primary Key:** (requestedBillingNumber, requestedLineNumber, requestedServiceNumber) **Foreign Key:** requestedServiceNumber **Foreign Key:** (requestedBillingNumber, requestedLineNumber)	**RoutineRoomService**(routineAssignedTimestamp, routineServiceNumber) **Primary Key:** routineServiceNumber **Foreign Key:** routineServiceNumber
CustomerServiceRequest(requestedCustomerNum, customerRequestedBillingNumber, customerRequestedLineNumber, customerRequestedServiceNumber, requestTimestamp) **Primary Key:** (requestedCustomerNum, customerRequestedBillingNumber, customerRequestedLineNumber, customerRequestedServiceNumber) **Foreign Key:** requestedCustomerNum **Foreign Key:** (customerRequestedBillingNumber, customerRequestedLineNumber, customerRequestedServiceNumber)	**RoomServiceAction**(roomServiceNumber, serviceName, roomServiceDescription) **Primary Key:** roomServiceNumber
Cleaning(cleaningType, cleaningRoomServiceNumber)	**AddingSupply**(addingSupplyRoomServiceNumber)

Primary Key: cleaningRoomServiceNumber **Foreign Key:** cleaningRoomServiceNumber	**Primary Key:** addingSupplyRoomServiceNumber **Foreign Key:** addingSupplyRoomServiceNumber
RequestedPerformedRoomServiceAction(performRequestedBillingNumber, performRequestedLineNumber, performRequestedServiceNumber, performRequestedRoomServiceNumber) **Primary Key:** (performRequestedBillingNumber, performRequestedLineNumber, performRequestedServiceNumber, performRequestedRoomServiceNumber) **Foreign Key:** performRequestedRoomServiceNumber **Foreign Key:** (performRequestedBillingNumber, performRequestedLineNumber, performRequestedServiceNumber)	**RoutinePerformedRoomServiceAction**(performRoutineServiceNumber, performRoutineRoomServiceNumber) **Primary Key:** (performRoutineServiceNumber, performRoutineRoomServiceNumber) **Foreign Key:** performRoutineServiceNumber **Foreign Key:** performRoutineRoomServiceNumber
ReservedRoom(reservedRoomReservationNumber, reservedRoomBranchNumber, reservedRoomNumber) **Primary Key:** (reservedRoomReservationNumber, reservedRoomBranchNumber, reservedRoomNumber) **Foreign Key:** reservedRoomReservationNumber **Foreign Key:** (reservedRoomBranchNumber, reservedRoomNumber)	**PropertyInspection**(propertyInspectionNumber, propertyInspectionTimestamp, propertyInspectionDescriptions, propertyInspectionEmployeeNum, propertyInspectionRentalNumber, propertyInspectionBranchNumber, propertyInspectionRoomNumber) **Primary Key:** propertyInspectionNumber **Foreign Key:** propertyInspectionEmployeeNum **Foreign Key:** propertyInspectionRentalNumber **Foreign Key:** (propertyInspectionBranchNumber, propertyInspectionRoomNumber)
Vendor(vendorNumber, vendorName, vendorType, vendorBuildingNumber, vendorStreetName, vendorSubdistrict, vendorDistrict, vendorProvince, vendorPostalCode, vendorEmail, vendorEmployeeNum, vendorCreatedTimestamp) **Primary Key:** vendorNumber **Foreign Key:** vendorEmployeeNum	**VendorTelephone**(vendorTelephoneNumber,telephoneVendorNumber) **Primary Key:** vendorTelephoneNumber Foreign Key: telephoneVendorNumber
InventoryEntry(entryNumber, entryDescription, supplyUnitQuantity, overseeingTimestamp, inventoryEntryType, entrySupplyNumber, entryReturningSupplyNumber, entryObjectNumber, entryAccountantNumber, entryServiceNumber, entryBranchNumber) **Primary Key:** entryNumber **Foreign Key:** entrySupplyNumber **Foreign Key:** entryAccountantNumber **Foreign Key:** entryServiceNumber **Foreign Key:** entryBranchNumber **Foreign Key:** entryReturningSupplyNumber **Foreign Key:** entryObjectNumber	**Purchasing**(purchasingNumber, purchasingTimeCreated, purchasingTimeApproved, purchasingTimeRejected, purchasingTimePurchased, purchasingType, purchasingVendorNumber, purchasingManagerNum, purchasingAccountantNum, purchasingRequestorEmployeeNum, purchasingFinancialTransactionNumber) **Primary Key:** purchasingNumber **Foreign Key:** purchasingVendorNumber **Foreign Key:** purchasingManagerNum **Foreign Key:** purchasingAccountantNum **Foreign Key:** purchasingRequestorEmployeeNum **Foreign Key:** purchasingFinancialTransactionNumber
PurchasingLine(linePurchasingNumber, purchasingLineNumber, purchasingLineTimeAdded, purchasingLinePricePerUnit, purchasingLineQuantity, purchasingLineDiscount, purchasingLineDiscountDescription, purchasingLineSupplyNumber, purchasingLineFinancialTransactionNumber, purchasingLineInventoryEntryNumber)	**PurchasingEmployee**(purchasingEmployeePurchasingNumber, purchasingEmployeeNum) **Primary Key:** (purchasingEmployeePurchasingNumber, purchasingEmployeeNum) **Foreign Key:** (purchasingEmployeePurchasingNumber **Foreign Key:** purchasingEmployeeNum

Primary Key: (linePurchasingNumber, purchasingLineNumber) **Foreign Key:** linePurchasingNumber **Foreign Key:** purchasingLineSupplyNumber **Foreign Key:** purchasingLineFinancialTransactionNumber **Foreign Key:** purchasingLineInventoryEntryNumber	
RoomObjectModel(objectModelId, objectModelBarcode, objectModelCategory, objectModelMaterial, objectMainColor) **Primary Key:** objectModelId	**RoomObject**(objectNumber, objectDateMovedIn, objectBranchNumber, objectRoomNumber, roomObjectModelId) **Primary Key:** objectNumber **Foreign Key:** roomObjectModelId **Foreign Key:** (objectBranchNumber, objectRoomNumber)
PropertyDamage(propertyDamageNumber, damageDescription, damageSeverity, propertyDamageInspectionNumber, propertyDamageBillingNumber, propertyDamageLineNumber, propertyDamageObjectNumber, propertyDamageMaintenanceNumber, propertyDamageManagerNum, propertyDamageApprovedTimestamp, propertyDamageRequestorEmplNum) **Primary Key:** propertyDamageNumber **Foreign Key:** propertyDamageInspectionNumber **Foreign Key:** propertyDamageObjectNumber **Foreign Key:** propertyDamageMaintenanceNumber **Foreign Key:** propertyDamageManagerNum **Foreign Key:** propertyDamageRequestorEmplNum **Foreign Key:** (propertyDamageBillingNumber, propertyDamageLineNumber)	**ExternalOperative**(extOperativeNum, extOperativeFirstName, extOperativeMiddleName, extOperativeLastName) **Primary Key:** extOperativeNum
ExternalOperativePhone(extOperativeNumber,phoneExtOperativeNumber) **Primary Key:** extOperativeNumber **Foreign Key:** phoneExtOperativeNumber	**MaintenanceExternalOperative**(maintenanceExtOperativeNum, extOperativemaintenanceNumber) **Primary Key:** (maintenanceExtOperativeNum, extOperativemaintenanceNumber) **Foreign Key:** maintenanceExtOperativeNum **Foreign Key:** extOperativemaintenanceNumber
MaintenanceRoomObject(maintenanceRoomObjectNumber, roomMaintenanceNumber) **Primary Key:** (maintenanceRoomObjectNumber, roomMaintenanceNumber) **Foreign Key:** maintenanceRoomObjectNumber **Foreign Key:** roomMaintenanceNumber	**MaintenanceEmployee**(maintenanceEmployeeNum, employeeMaintenanceNumber) **Primary Key:** (maintenanceEmployeeNum, employeeMaintenanceNumber) **Foreign Key:** maintenanceEmployeeNum **Foreign Key:** employeeMaintenanceNumber
EmployeeWagePayment(wagePaymentFinancialTransactionNumber, wagePaymentManagerNum, wagePaymentEmployeeNum) **Primary Key:** (wagePaymentFinancialTransactionNumber, wagePaymentManagerNum, wagePaymentEmployeeNum) **Foreign Key:** wagePaymentFinancialTransactionNumber **Foreign Key:** wagePaymentManagerNum **Foreign Key:** wagePaymentEmployeeNum	**MaintenanceSupplyRequirement**(maintenanceSupplyInventoryEntryNumber, supplyMaintenanceNumber, maintenanceSupplyNumber, remarks) **Primary Key:** (maintenanceSupplyInventoryEntryNumber, supplyMaintenanceNumber, maintenanceSupplyNumber) **Foreign Key:** maintenanceSupplyInventoryEntryNumber

	Foreign Key: supplyMaintenanceNumber **Foreign Key:** maintenanceSupplyNumber
InventoryEntryRecorder(recordInventoryEntryNumber, recordEmployeeNum, entryRecordTimestamp) **Primary Key:** (recordInventoryEntryNumber, recordEmployeeNum) **Foreign Key:** recordInventoryEntryNumber **Foreign Key:** recordEmployeeNum	**CleaningServiceReturningSupply**(cleaningServiceReturningSupplyNumber, cleaningRoomServiceNumber) **Primary Key:** (cleaningServiceReturningSupplyNumber, cleaningRoomServiceNumber) **Foreign Key:** cleaningServiceReturningSupplyNumber **Foreign Key:** cleaningRoomServiceNumber
AddingSupplyServiceSupply(addingServiceSupplyNumber, addingRoomServiceNumber) **Primary Key:** (addingServiceSupplyNumber, addingRoomServiceNumber) **Foreign Key:** addingServiceSupplyNumber **Foreign Key:** addingRoomServiceNumber	

Table 2 – List of all relations of the logical model of the Bright House database

Transaction Analysis

ORIGINAL DELIVERABLE REQUIREMENT

d. The results of the analysis of at least two transactions for physical design (see Table 18.1 and Figure 18.4 as examples).

To clearly understand the usage of the transaction occurred in the database, an analysis of transactions in the physical database is required. From the first phase or the first section of this report, there are several transactions for the Bright House database. In this section, we have selected 4 important transactions that are frequently used by stakeholders.

1. **P5: Insert the details of a new Customer Payment Billing line**
   This is a procedure that facilitates a consistent insertion of a payment billing line that relates to the customer. This procedure is used by many other stored procedures because there are multiple circumstances that can lead to customer debt or billing. For example, new room service is requested; thus, a new billing line has to be inserted. Another example would be when a customer starts renting a room, a new line for customer billing has to be added for the rental fee of that customer.
2. **V9: List Average Customer Payment grouped by Month and Customer's nationality**
   This is a view that facilitates the performance of the business by observing the average amount of money in each customer transaction every month in every year. Directors and apartment managers mainly use it.
3. **V12: For each branch, retrieve information regarding rentals and customers such as rental duration, PAX, etc. Then summarize them with the average or total sum.**
   This is a view that facilitates branch managers by displaying the detailed information of each branch in terms of rentals. For each branch, total rentals, total customers, total unique customers, average rental duration in days, and average PAX (number of customers per rental) will be summarized.
4. **V15: List room number, customer name, rental type, deposit, rental amount, as well as supplies used by the room.**

This is a view that lists each rental in Bright House apartment in deeper detail. Employees can use this detail to determine the service as well as determine the capacity that the branch can accommodate given a time. Moreover, each rental will be listed in terms of the identification number, rental staring time, the starting time of the first rental period, branch number, room number, number of customers, total billing amount, deposit, list of supplies used, as well as its count.

Transaction/Relation Cross-Reference Matrix

To show the way each relation is associated with transactions, we will use the matrix below. In the following pages, we will focus on the four transactions previously listed.

No.	Relations	Transaction P5				Transaction V9				Transaction V12				Transaction V15			
		I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
1	Accountant																
2	AddingSupply																
3	AddingSupplyServiceSupply																
4	Branch										×						
5	BranchTelephone																
6	Cleaning																
7	CleaningPersonnel																
8	CleaningServiceReturningSupply																
9	Customer						×										
10	CustomerBilling	×	×				×								×		
11	CustomerBillingLine	×	×				×								×		
12	CustomerPassport																
13	CustomerRoomService																
14	CustomerServiceRequest														×		
15	CustomerTelephone																
16	CustomerVisa																
17	Employee																
18	EmployeeDailyWorkHours																
19	EmployeeTelephone																
20	EmployeeWagePayment																
21	EmployeeWorkingShift																
22	ExternalOperative																
23	ExternalOperativePhone																
24	Feedback																
25	FeedbackEmployeeAction																
26	FinancialTransaction		×														

No.	Relations	Transaction P5				Transaction V9				Transaction V12				Transaction V15			
		I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
27	InventoryEntry														×		
28	InventoryEntryRecorder																
29	Maintenance																
30	MaintenanceEmployee																
31	MaintenanceExternalOperative																
32	MaintenanceRoomObject																
33	MaintenanceSupplyRequirement																
34	Manager																
35	PropertyDamage																
36	PropertyInspection																
37	Purchasing																
38	PurchasingEmployee																
39	PurchasingLine																
40	Rental										×				×		
41	RentalCustomer										×				×		
42	RentalPeriod														×		
43	RequestedPerformedRoomServiceAction																
44	RequestedRoomService																
45	Reservation																
46	ReservedRoom																
47	ReturningSupply																
48	Room										×						
49	RoomObject																
50	RoomObjectModel																
51	RoomServiceAction																
52	RoomUtilityRecord																
53	RoutinePerformedRoomServiceAction																
54	RoutineRoomService																
55	Supply																
56	UtilityCost																
57	UtilityCostOperand																
58	Vendor																
59	VendorTelephone																

Table 3 – Matrix showing the cross-references between the four important transactions and relations in the database.

Remark: I: Insert, R: Read, U: Update, D: Delete

Overview

The figure below is the transaction usage map for the transaction P5, V9, V12, and V15 in the previous topics. Each table is also labeled with the initial size accordingly with phase 3 of the project. Each emphasized transaction is highlighted with a distinct color. Please note that the transaction V15 has multiple paths due to its complicated join and join with a derived table. Transaction V12's transaction path is cyclic because there are 2 join clauses that involve with the Rental table.

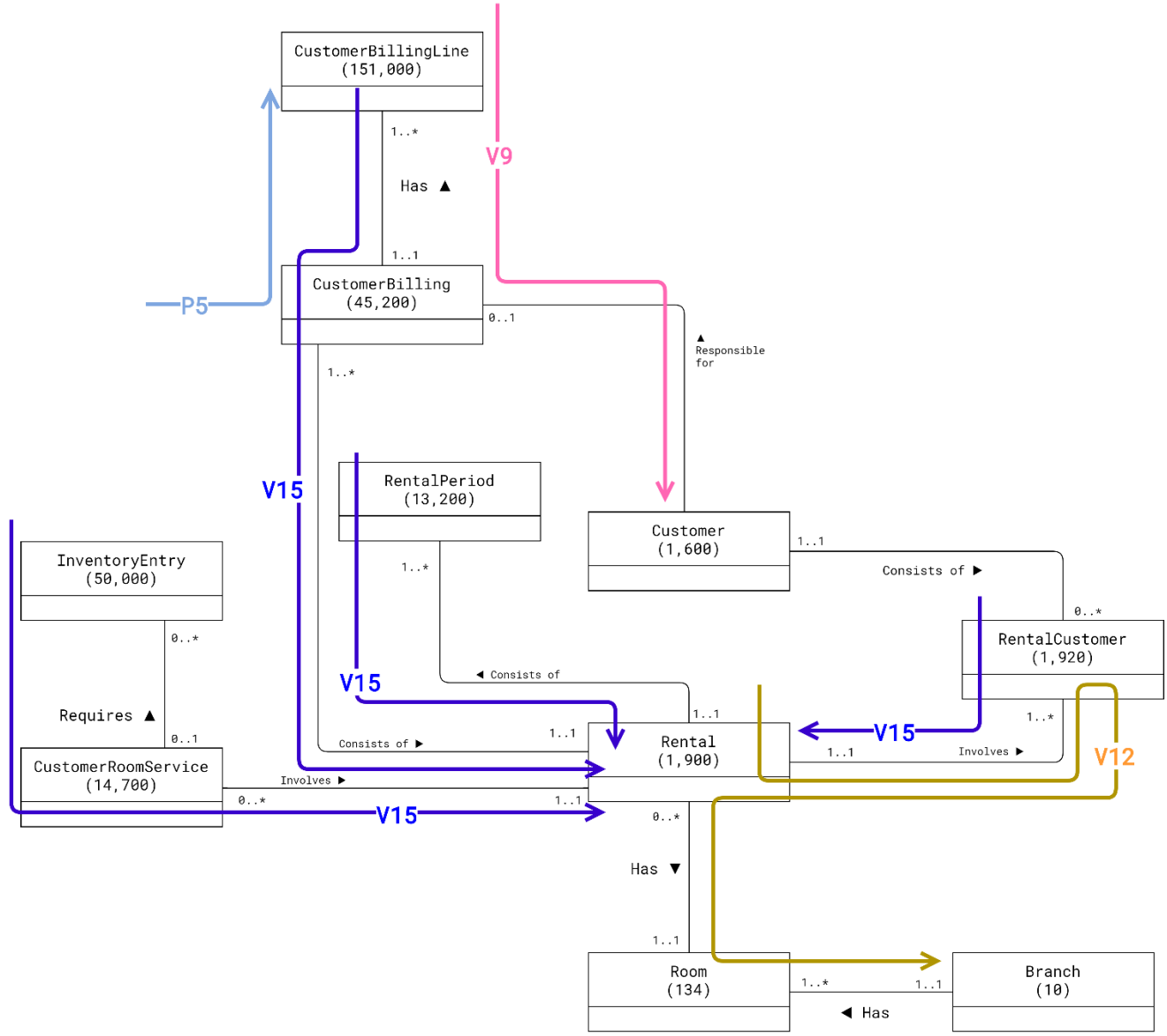


Figure 2 – Overview of the paths of the transactions that will be analyzed in the next section.

Transaction Analysis Forms

In this section, the four important transactions will be analyzed in detail using the transaction analysis form. Please note that each transaction has assumptions on the order of the database table access.

Form 1 - Analysis of transaction P5: Insert the details of a new Customer Payment Billing line

Example: *Insert a customer billing line of a transaction based on a utility cost of a given billing number.*

Transaction Analysis Form
29 April 2020

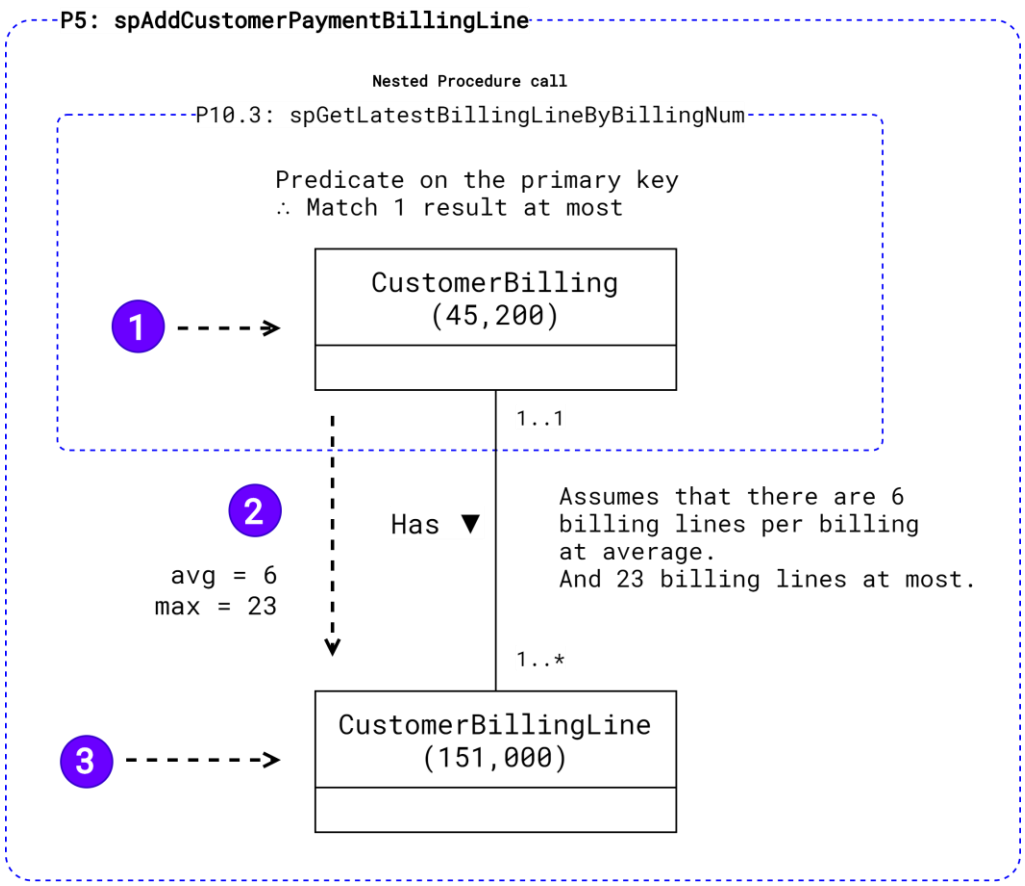
Transaction	**P5**: Insert the details of a new Customer Payment Billing line
Transaction Volume	
Average:	**2 per hour** (Assume that there will be ~30 new billing lines per day)
Peak:	**3 per hour** (Between 15:00 and 18:00 Every day)

```
/**
 * -- 5: Insert the details of a new Customer Payment Billing line
 *
 * This method defines a way for other procedures to add/insert a new billing line.
 */
CREATE PROCEDURE spAddCustomerPaymentBillingLine(
   @targetBillingNum int,
   @lineName varchar(40),
   @lineAmount decimal(19, 4),
   @lineStatus varchar(10),
   @lineDiscountAmount decimal(19,4),
   @lineRecorderEmployeeNum int,
   @outputBillingNum int OUTPUT,
   @outputBillingLineNum int OUTPUT
)
AS
BEGIN
   SET NOCOUNT ON

   -- Get the latest Billing Line Associated with the billing number
   DECLARE @targetBillingLineTable table(targetBillingLineNum int)
   INSERT @targetBillingLineTable EXEC spGetLatestBillingLineByBillingNum @targetBillingNum

   -- Get the last billing line of the given target billing number
   DECLARE @targetBillingLine int
   SELECT @targetBillingLine = targetBillingLineNum FROM @targetBillingLineTable

   -- Sometimes the @targetBillingLine may already equal to ONE
   IF NOT @targetBillingLine = 1
       SET @targetBillingLine = @targetBillingLine + 1

   -- If we cannot find the last billing line, it means that it does not exist.
   IF @targetBillingLine IS NULL
```

```
       THROW 50001, 'Sorry, we cannot insert a new Billing Line because we cannot find a
BillingLine number within the rental period.', 1

   -- Current Time
   DECLARE @currentTime datetime = CURRENT_TIMESTAMP

   -- Insert a billing line into the associated main bill
   INSERT INTO CustomerBillingLine
   VALUES (@targetBillingNum, @targetBillingLine, @lineName, @lineAmount, @lineStatus,
@currentTime,
           @lineDiscountAmount, @lineRecorderEmployeeNum, @currentTime)

   -- The Query returns some value
   SELECT @outputBillingNum = customerBillingNumber, @outputBillingLineNum =
customerBillingLineNumber
   FROM CustomerBillingLine
   WHERE customerBillingNumber = @targetBillingNum AND customerBillingLineNumber =
@targetBillingLine

   -- Select the newly inserted item.
   SELECT *
   FROM CustomerBillingLine
   WHERE customerBillingNumber = @targetBillingNum AND customerBillingLineNumber =
@targetBillingLine
END
GO;
```

Predicate:	○ billingNumber *(From **EXEC** spGetLatestBillingLineByBillingNum)* ○ customerBillingNumber ○ customerBillingLineNumber
Join attributes:	○ B.billingNumber = CBL.customerBillingNumber *(From **EXEC** spGetLatestBillingLineByBillingNum)*
Ordering attribute:	○ *NONE*
Grouping attribute:	○ billingNumber *(From **EXEC** spGetLatestBillingLineByBillingNum)*
Built-in functions:	○ *MAX*(customerBillingLineNumber) *(From **EXEC** spGetLatestBillingLineByBillingNum)*
Attributes updated:	○ customerBillingNumber ○ customerBillingLineNumber ○ customerBillingLineItemName ○ customerBillingAmount ○ customerBillingStatus ○ custoemrBillingTime ○ customerBillingDiscountAmount ○ customerBillingEmployeeId ○ customerBillingCreatedTimestamp

Transaction usage map



Access	Entity	Type of Access	No. of References		
			Per Transaction	Avg Per Hour	Peak Per Hour
1	CustomerBilling	Read	45,200*	90,400	135,600
2	CustomerBillingLine	Read	6 - 23	12 - 46	18 - 184
3	CustomerBillingLine	Insert	1	2	3
Total References			45,207 – 45,224	90,414 – 90,448	135,621 – 135,787

- **Access 1:** Assume that we have to search according to the predicate by accessing all rows of the `CustomerBilling`.
- **Access 2:** The max and average are based on the data loaded into the database.
- A Nested procedure call (e.g. `spGetLatestBillingLineByBillingNum`) is also taken into account.

Form 2 - Analysis of transaction V9: List Average Customer Payment grouped by Month and Customer's nationality

Example: *Identify the average amount of customer payments grouped by month and customer's nationality.*

Transaction Analysis Form
29 April 2020

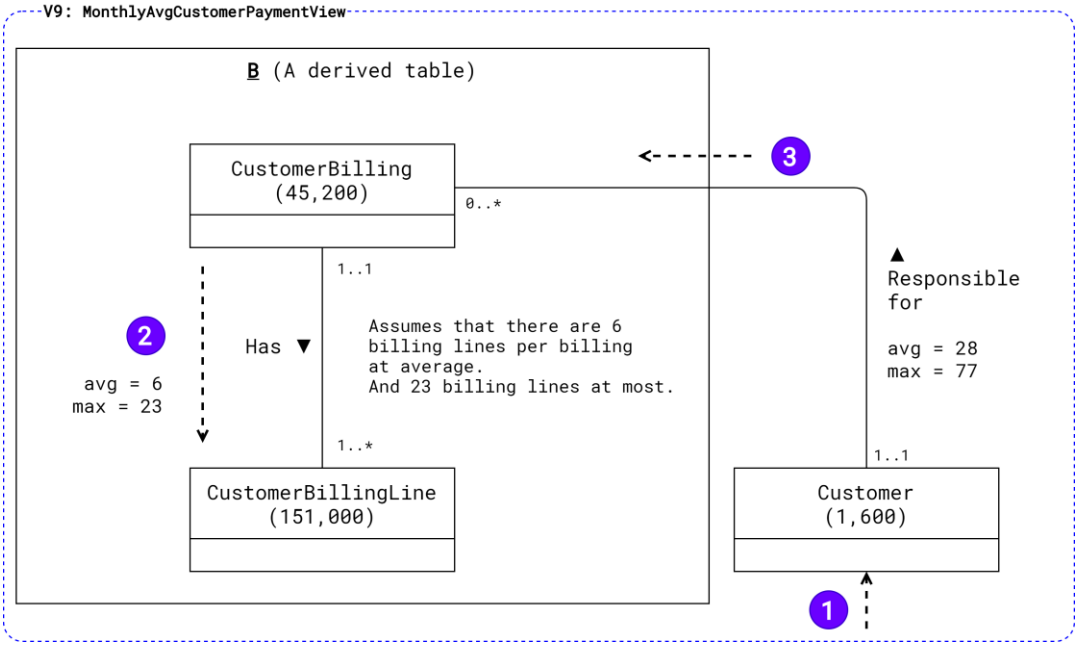
Transaction	**V9**: List Average Customer Payment grouped by Month and Customer's nationality
Transaction Volume	
Average:	**30 per hour** (Assume this view is used by the directors and every branch manager via a managerial dashboard that automatically refreshes data)
Peak:	**35 per hour** (Between 15:00 and 18:00 Friday)

```
/**
* -- 9: List Average Customer Payment grouped by Month and Customer's nationality
*
* Identify the average amount of customer payments grouped by month and customer's nationality.
*/
CREATE VIEW MonthlyAvgCustomerPaymentView AS
SELECT month, year, C.customerNationality, AVG(billingAmount) AS averagePaymentAmount
FROM (
    SELECT customerBillingNumber,
           billingCustomerNum,
           SUM(customerBillingAmount) AS billingAmount,
           MONTH(customerBillingTime) AS month,
           YEAR(customerBillingTime)  AS year
    FROM CustomerBilling
             JOIN CustomerBillingLine CBL ON CustomerBilling.billingNumber =
CBL.customerBillingNumber
    GROUP BY customerBillingNumber, customerBillingTime, billingCustomerNum
    ) B
        JOIN Customer C ON C.customerNum = B.billingCustomerNum
GROUP BY C.customerNationality, month, year;
```

Predicate:	○ *NONE*
Join attributes:	○ C.customerNum = B.billingCustomerNum ○ CustomerBilling.billingNumber = CBL.customerBillingNumber
Ordering attribute:	○ *NONE*
Grouping attribute:	○ customerBillingNumber ○ customerBillingTime

	○ billingCustomerNum ○ customerNationality ○ customerBillingTime
Built-in functions:	○ SUM(customerBillingAmount) **AS billingAmount** ○ MONTH(customerBillingTime) **AS month** ○ YEAR(customerBillingTime) **AS year** ○ AVG(billingAmount) **AS averagePaymentAmount**
Attributes updated:	○ *NONE*

Transaction usage map



Access	Entity	Type of Access	No. of References		
			Per Transaction*	Avg Per Hour	Peak Per Hour
1	Customer	Read	1,600	48,000	56,000
2	CustomerBilling	Read	44,800 - 123,200	1,344,000 - 3,696,000	1,568,000 - 4,312,000
3	CustomerBillingLine	Read	268,800 - 1,030,400	8,062,000 - 30,912,000	9,408,000 - 36,064,000
Total References			315,200 - 1,155,200	9,454,000 - 34,656,000	11,032,000 - 40,432,000

- A derived table named “B” is also considered and visualized using a rectangle with a label.

Form 3 - Analysis of transaction V12: For each branch, retrieve information regarding rentals and customers such as rental duration, PAX, etc. Then summarize them with the average or total sum.

Example: *Identify the total rentals, total customers, total unique customers, average rental duration in days and average PAX of the branch number 2*

Transaction Analysis Form
29 April 2020

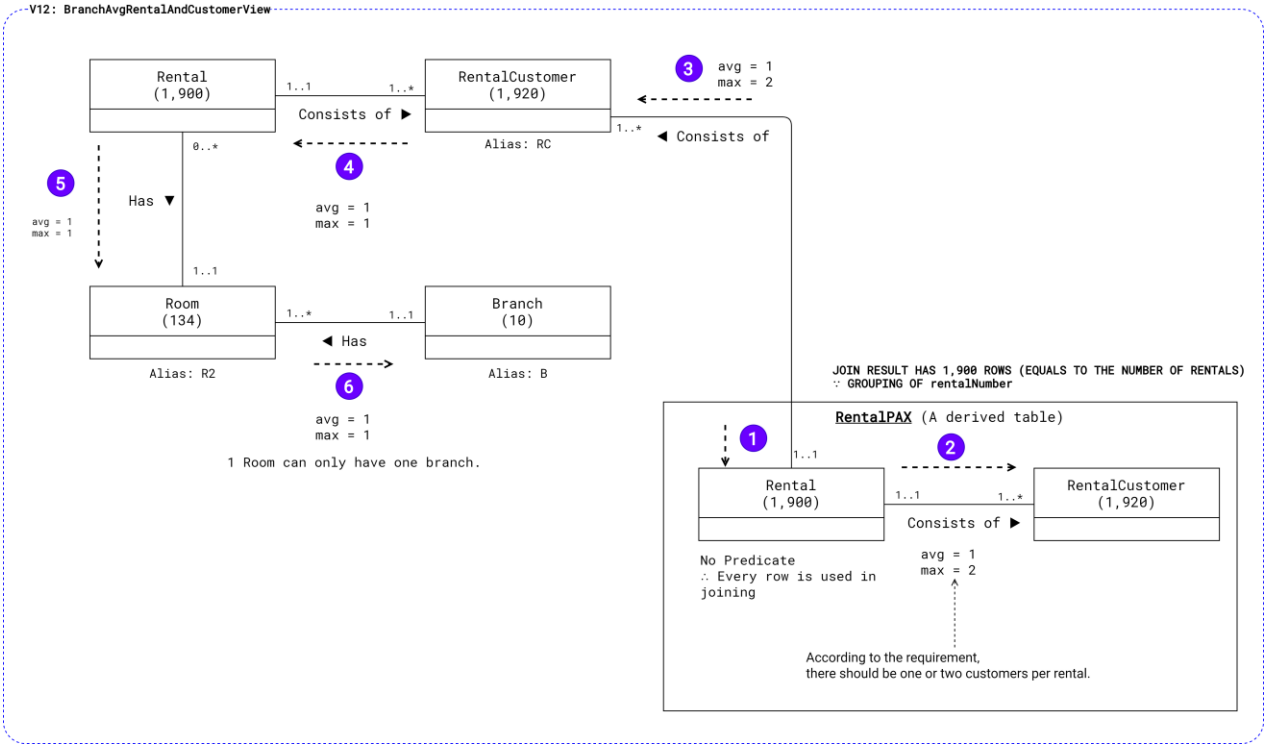
Transaction	**V12**: For each branch, retrieve information regarding rentals and customers such as rental duration, PAX, etc. Then summarize them with the average or total sum.
Transaction Volume	
Average:	**30 per hour** (Assume this view is used by the owner and every branch manager via a managerial dashboard that automatically refreshes data)
Peak:	**35 per hour** (Between 15:00 and 18:00 Friday)

```sql
/**
* -- 12: For each branch, retrieve information regarding rentals and customers such as rental
duration, PAX, etc.
* Then summarize them with average or total sum.
*/
CREATE VIEW BranchAvgRentalAndCustomerView
AS
SELECT branchNumber,
       COUNT(DISTINCT rentalNumber)                                  AS totalRentals,
       COUNT(ALL rentalCustomerNum)                                  AS totalCustomers,
       COUNT(DISTINCT rentalCustomerNum)                             AS totalUniqueCustomers,
       AVG(DATEDIFF(DAY, rentalStartingTime, rentalEndingTime)) AS averageRentalDurationDay,
       AVG(PAX)                                                      AS averagePAX
FROM (
    SELECT branchNumber, rentalNumber, rentalCustomerNum, rentalStartingTime, rentalEndingTime, PAX
    FROM Branch B
         JOIN Room R2 ON B.branchNumber = R2.roomBranchNumber
         JOIN Rental ON R2.roomBranchNumber = Rental.rentalBranchNumber AND R2.roomNumber =
Rental.rentalRoomNumber
         JOIN RentalCustomer RC ON Rental.rentalNumber = RC.rentalCustomerRentalNumber
         JOIN (
              -- To find the average PAX, we must count them by each rental first.
             SELECT rentalNumber AS rentalNum, COUNT(rentalCustomerNum) AS PAX
             FROM Rental
                      JOIN RentalCustomer C ON Rental.rentalNumber = C.rentalCustomerRentalNumber
             GROUP BY rentalNumber
         ) RentalPAX ON RentalPAX.rentalNum = RC.rentalCustomerRentalNumber) A
GROUP BY branchNumber
```

Predicate:	○ *NONE*
Join attributes:	○ Rental.rentalNumber = C.rentalCustomerRentalNumber

	○ Rental.rentalBranchNumber = R2.roomBranchNumber **AND** Rental.rentalRoomNumber = R2.roomNumber ○ Rental.rentalNumber = RC.rentalCustomerRentalNumber ○ R2.roomBranchNumber = B.branchNumber ○ RentalPAX.rentalNum = RC.rentalCustomerRentalNumber
Ordering attribute:	○ *NONE*
Grouping attribute:	○ *rentalNumber* ○ *branchNumber*
Built-in functions:	○ COUNT(rentalCustomerNum) **AS** PAX ○ COUNT(**DISTINCT** rentalNumber) **AS** totalRentals ○ COUNT(**ALL** rentalCustomerNum) **AS** totalCustomers ○ COUNT(**DISTINCT** rentalCustomerNum) **AS** totalUniqueCustomers ○ AVG(DATEDIFF(**DAY**, rentalStartingTime, rentalEndingTime)) **AS** averageRentalDurationDay ○ AVG(PAX) **AS** averagePAX
Attributes updated:	○ *NONE*

Transaction usage map



Access	Entity	Type of Access	No. of References		
			Per Transaction*	Avg Per Hour	Peak Per Hour
1	Rental	Read	1,900	57,000	66,500
2	RentalCustomer	Read	1,900 - 3,800	57,000 - 114,000	66,500 - 133,000
3	RentalCustomer	Read	1,900 - 7,600	57,000 - 228,000	66,500 - 266,000
4	Rental	Read	1,900 - 7,600	57,000 - 228,000	66,500 - 266,000
5	Room	Read	1,900 - 7,600	57,000 - 228,000	66,500 - 266,000
6	Branch	Read	1,900 - 7,600	57,000 - 228,000	66,500 - 266,000
Total References			11,400 - 36,100	342,000 - 627,000	399,000 - 1,263,500

- The result of the derived table (e.g. **`RentalPAX`**) has the number of rows equals the number of rows in the **`Rental`** record. This is because of the **`GROUP BY`** clause on `rentalNumber`.
- **In access numbers 3, 4, 5, and 6**, The numbers of references seem to be fixed. This is because the join clauses in this view are supposed to append more relevant data columns into the result; therefore, the number of rows is unchanged.

Form 4 - Analysis of transaction V15: List room number, customer name, rental type, deposit, rental amount, as well as supplies used by the room.

Example: *List the details of rental numbers 1, 2, and 100 in terms of supply usage as well as its number, starting time, the period starting time, branch, room number, PAX, billing amount, and deposit.*

Transaction Analysis Form
29 April 2020

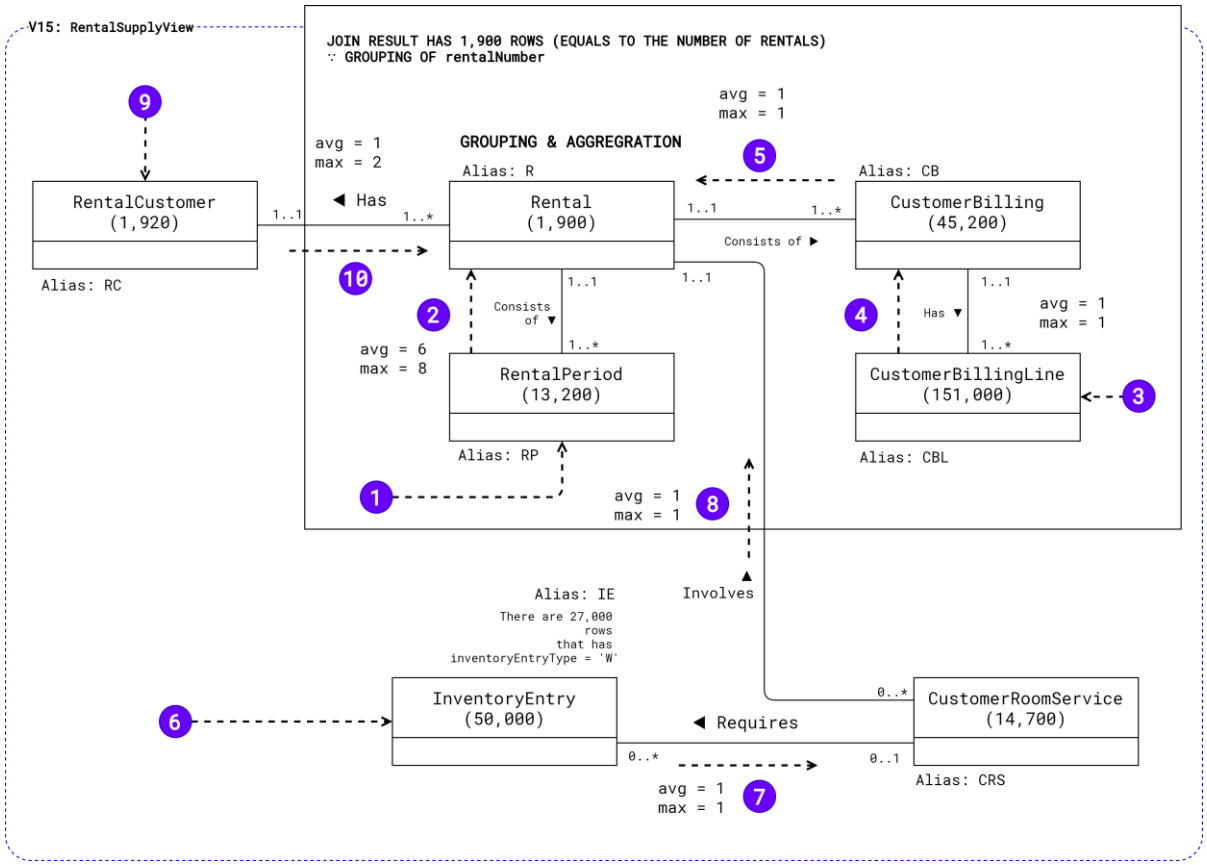
Transaction	**V15**: List room number, customer name, rental type, deposit, rental amount, as well as supplies used by the room.
Transaction Volume	
Average:	30 per hour (Assume that every employee can access this View)
Peak:	35 per hour

```sql
/**
* -- 15: List room number, customer name, rental type, deposit, rental amount, as well as supplies used by the room.
*/
CREATE VIEW RentalSupplyView
AS
SELECT A.rentalNumber,
      rentalStartingTime,
      periodStartTime,
      rentalBranchNumber       AS branch,
      rentalRoomNumber         AS roomNumber,
      COUNT(rentalCustomerNum) AS PAX,
      totalBillingAmount,
      rentalDeposit,
      XZ.supplyNum,
      SUM(XZ.quantity)         AS supplyCount
FROM (
    SELECT rentalNumber,
           rentalStartingTime,
           MAX(rentalPeriodStartingTime) AS periodStartTime,
           rentalBranchNumber,
           rentalRoomNumber,
           SUM(customerBillingAmount)    AS totalBillingAmount,
           rentalDeposit
    FROM Rental R
             JOIN CustomerBilling CB ON R.rentalNumber = CB.billingRentalNumber
             JOIN CustomerBillingLine CBL ON CB.billingNumber = CBL.customerBillingNumber
             JOIN RentalPeriod RP ON R.rentalNumber = RP.rentalPeriodRentalNumber
    GROUP BY rentalNumber, rentalStartingTime, rentalBranchNumber, rentalRoomNumber, rentalDeposit
    ) A
        JOIN RentalCustomer RC ON A.rentalNumber = RC.rentalCustomerRentalNumber
```

```
        JOIN (
             SELECT serviceRentalNumber,
                    COALESCE(entrySupplyNumber, entryReturningSupplyNumber) AS supplyNum,
                    SUM(supplyUnitQuantity)                                 AS quantity
             FROM InventoryEntry IE
                      JOIN CustomerRoomService CRS ON IE.entryServiceNumber = CRS.serviceNumber
             WHERE (inventoryEntryType = 'W' AND entrySupplyNumber IS NOT NULL)
               OR (inventoryEntryType = 'W' AND entryReturningSupplyNumber IS NOT NULL)
             GROUP BY serviceRentalNumber, entrySupplyNumber, entryReturningSupplyNumber) XZ
             ON XZ.serviceRentalNumber = rentalNumber
GROUP BY A.rentalNumber, rentalStartingTime, periodStartTime, rentalBranchNumber, rentalRoomNumber,
totalBillingAmount,
        rentalDeposit, XZ.supplyNum;
```

Predicate:	○ *inventoryEntryType = 'W' AND (entrySupplyNumber IS NOT NULL OR entryReturningSupplyNumber IS NOT NULL)*
Join attributes:	○ R.rentalNumber = CB.billingRentalNumber ○ CB.billingNumber = CBL.customerBillingNumber ○ R.rentalNumber = RP.rentalPeriodRentalNumber ○ A.rentalNumber = RC.rentalCustomerRentalNumber ○ IE.entryServiceNumber = CRS.serviceNumber ○ XZ.serviceRentalNumber = rentalNumber
Ordering attribute:	○ NONE
Grouping attribute:	○ rentalNumber ○ rentalStartingTime ○ rentalBranchNumber ○ rentalRoomNumber ○ rentalDeposit ○ serviceRentalNumber ○ entrySupplyNumber ○ entryReturningSupplyNumber ○ *periodStartTime* ○ *totalBillingAmount*
Built-in functions:	○ MAX(rentalPeriodStartingTime) AS periodStartTime ○ SUM(customerBillingAmount) AS totalBillingAmount ○ COALESCE(entrySupplyNumber, entryReturningSupplyNumber) AS supplyNum ○ SUM(supplyUnitQuantity) AS quantity ○ COUNT(rentalCustomerNum) AS PAX ○ SUM(XZ.quantity) AS supplyCount
Attributes updated:	○ *NONE*

Transaction usage map



Access	Entity	Type of Access	No. of References		
			Per Transaction	Avg Per Hour	Peak Per Hour
1	RentalPeriod	Read	13,200	396,000	462,000
2	Rental	Read	79,200 - 105,600	2,376,000 - 3,168,000	2,772,000 - 3,696,000
3	CustomerBillingLine	Read	151,000	4,530,000	5,285,000
4	CustomerBilling	Read	151,000	4,530,000	5,285,000
5	Rental	Read	151,000	4,530,000	5,285,000
6	InventoryEntry	Read	50,000	1,500,000	1,750,000
7	CustomerRoomService	Read	27,000	810,000	945,000
8	Rental	Read	27,000	810,000	945,000
9	RentalCustomer	Read	1,920	57,600	67,200
10	Rental	Read	1,920 - 3,840	57,600 - 115,200	67,200 - 134,400
Total References			653,240 - 681,560	19,597,200 - 20,446,800	22,863,400 - 23,854,600

No. of References (nRef) Calculation (Per transaction)

Access	Calculation	Result (No. of References)
1	**nRecord of RentalPeriod**	**13,200**
2	[avg × **Y**, max × **Y**] **Given that Y = nRecord of RentalPeriod**	[6 × 13200, 8 × 13200] = **[79200, 105600]**
3	**nRecord of CustomerBillingLine**	**151,000**
4	[avg × **A**, max × **A**] **Given that A = nRecord of CustomerBillingLine**	[1 × 151000, 1× 151000] = **[**151000, 151000**] = 151,000**
5	[avg × **R**, max × **R**] **Given that R = avg nRef from Access #4**	[1 × 151000, 1× 151000] = **[**151000, 151000**] = 151,000**
6	**nRecord of InventoryEntry**	**50,000**
7	[avg × **Q**, max × **Q**] **Given that Q** = nRecord of InventoryEntry WHERE inventoryEntryType='W'	[1 × 27000, 1 × 27000] = [27000, 27000] = **27,000**
8	[avg × **M**, max × **M**] **Given that M = nRef from Access #7**	[1 × **27000**, 1× **27000**] = [27000, 27000] = **27,000**
9	**nRecord of RentalCustomer**	**1,920**
10	[avg × **G**, max × **G**] **Given that G = nRecord of RentalCustomer**	[1 × 1920, 2 × 1920] = **[1920, 3,840]**

Table 4 – Calculation details of No. of references per access

- The order of relation accesses is based on the query execution plan provided by Microsoft SQL Server Management Studio.

Index Analysis

ORIGINAL DELIVERABLE REQUIREMENT

e. The results of the analysis of at least two indexes for physical design (see page 579 – 581 and Table 18.2 – 18.4 as examples)

Interactions between relation and query transactions

After implementing some of the required transactions as listed in table 1, we continued by analyzing each transaction concerning its interaction with associated relations. We have identified join fields, ordering field, and grouping field so that we can later decide which are the indexes to create for each relation. Moreover, we also identified usage frequency per day for each field based on user usage context and requirements. Table 5 illustrates the analysis for every relation that satisfy conditions as follows:

1. Relations that have many data tuples [1]
2. Used by one of the implemented transactions in table 1

If a relation satisfies both conditions, it will be analyzed and listed in table 5.

No.	Relations	Transaction	Field	Frequency (per day)
4	Branch	(P1), (P16), (P24)	PREDICATE: branchNumber	5-40
		(P15)	PREDICATE: branchProvince	10 - 100
		(P15)	GROUPING: branchNumber, branchName, branchEmail, branchDistrict, branchProvince	10 - 100
		(V1)	GROUPING: branchNumber, branchName	3
		(V12)	GROUPING: branchNumber	3
9	Customer	(P3.1), (P21)	PREDICATE: customerNum	10 - 70
		(V7)	PREDICATE: customerGender	3
		(P17), (V3)	GROUPING: customerNum	5
		(P18)	GROUPING: customerNum, customerFirstName, customerMiddleName, customerLastName, customerDOB, customerEmail, customerNationality, customerProfession	10 - 30
		(P19)	GROUPING: customerNum, customerFirstName, customerMiddleName, customerLastName	10 - 40

No.	Relations	Transaction	Field	Frequency (per day)
		(V8), (V9)	GROUPING: customerNationality	10
		(V7)	GROUPING: customerNationality, customerDOB	3
10	CustomerBilling	(P10.3)	PREDICATE: billingNumber	10 - 80
		(P28), (V14)	JOIN: FinancialTransaction **on** financialTransactionNumber	8 - 40
		(V7), (V8), (V9), (V14)	JOIN: Customer **on** customerNum	6
		(P28), (V14), (V15)	JOIN: Rental **on** rentalNumber	8 - 50
		(P10.3)	GROUPING: billingNumber	10 - 80
		(P28)	GROUPING: billingNumber, billingDescription, billingCreatedTime	7 - 45
		(V7), (V8), (V9)	GROUPING: billingCustomerNum	6
11	CustomerBillingLine	(P5)	PREDICATE: customerBillingNumber	10 - 140
		(P5), (P6)	PREDICATE: customerBillingLineNumber	15 - 270
		(P10.3), (P28), (V7), (V8), (V9), (V15)	JOIN: CustomerBilling **on** billingNumber	23 - 130
		(V7), (V8)	GROUPING: customerBillingNumber	3
		(V8), (V9)	GROUPING: customerBillingNumber, customerBillingTime	3
12	CustomerPassport	(V3)	ORDERING: expirationDate	3 - 6
		(P3.1)	JOIN: Customer **on** customerNum	10 - 85
13	CustomerRoomService	(P10)	PREDICATE: customerRoomServiceNum	3
14	CustomerServiceRequest	(P10)	JOIN: RequestedRoomService on requestedBillingNumber **and** requestedLineNumber **and** requestedServiceNumber	3
15	CustomerTelephone	(P18)	JOIN: Customer **on** customerNum	10 - 80
16	CustomerVisa	(P3.2)	PREDICATE: visaPassportNumber	10 - 90
		(P3.2)	PREDICATE: visaNumber	10 - 90
		(V3)	ORDERING: arrivalDate	3 - 6
		(V3)	JOIN: CustomerPassport **on** passportNumber	3 - 6
17	Employee	(P9), (P15), (P16), (P24), (T2), (V1), (V17)	JOIN: Branch **on** branchNumber	40 - 410
		(P12.3), (P26), (P16)	PREDICATE: employeeBranchNumber	10

No.	Relations	Transaction	Field	Frequency (per day)
		(P12.2), (P16), (P26)	PREDICATE: employeeNum	10 - 20
		(P16)	PREDICATE: employeeFirstName	3
		(P16)	PREDICATE: employeeLastName	3
		(V1)	GROUPING: employeeNum, employeeFirstName, employeeMiddleName, employeeLastName, employeeDOB, employeeGender, employeeCitizenId	3
26	FinancialTransaction	(P26)	PREDICATE: financialTransactionAmount	3 - 10
26	InventoryEntry	(P12), (P14)	PREDICATE: entryNumber	10 - 90
		(P12.1), (V15)	PREDICATE: entryReturningSupplyNumber	20 - 140
		(V15)	PREDICATE: entrySupplyNumber	20 - 140
		(P12.1)	PREDICATE: entryBranchNumber	8
		(V4), (V5), (V15)	PREDICATE: inventoryEntryType	20 - 140
		(V4)	JOIN: Branch **on** branchNumber	3
		(V4), (V5)	JOIN: Supply **on** supplyNumber	5
		(V15)	JOIN: CustomerRoomService on serviceNumber	20 - 135
		(V4)	GROUPING: entryBranchNumber, entrySupplyNumber	3
		(V4)	GROUPING: entrySupplyNumber	3
		(V15)	GROUPING: entrySupplyNumber, entryReturningSupplyNumber	20 - 135
		(V5)	GROUPING: entryBranchNumber, entrySupplyNumber, overseeingTimestamp	3
29	Maintenance	(P24), (P25)	PREDICATE: maintenanceNumber	10 - 90
35	PropertyDamage	(P24), (V13)	JOIN: Maintenance **on** maintenanceNumber	10 - 65
		(V13)	JOIN: PropertyInspection **on** propertyInspectionNumber	3 - 5
		(P24)	PREDICATE: propertyDamageNumber	7 - 60
36	PropertyInspection	(P23), (P24)	PREDICATE: propertyInspectionNumber	15 - 150
		(V13)	JOIN: Rental **on** rentalNumber	3 - 5
37	Purchasing	(P14)	PREDICATE: purchasingNumber	10 - 75
		(P14)	PREDICATE: purchasingType	10 - 75
39	PurchasingLine	(P14)	PREDICATE: linePurchasingNumber	10 - 75

No.	Relations	Transaction	Field	Frequency (per day)
		(P14)	PREDICATE: purchasingLineNumber	10 - 75
		(P14)	JOIN: Purchasing **on** purchasingNumber	10 - 75
40	Rental	(P8), (P9), (P10.2), (P23), (P24), (P28), (T2)	PREDICATE: rentalNumber	40 - 640
		(P17), (P20), (V6)	PREDICATE: rentalStartingTime	20 - 120
		(P17), (P20), (V6)	PREDICATE: rentalEndingTime	20 - 120
		(P9), (T2)	JOIN: Branch **on** branchNumber	20 - 200
		(P17), (P19)	JOIN: Customer **on** customerNum	10 - 60
		(P20), (V6), (V12)	JOIN: Room **on** roomNumber **and** roomBranchNumber	10 - 120
		(P19)	GROUPING: rentalNumber, rentalRoomNumber, rentalBranchNumber, rentalEndingTime	10 - 55
		(P28), (V12)	GROUPING: rentalNumber	10 - 100
		(V10)	GROUPING: rentalStartingTime	5
		(V10)	GROUPING: rentalEndingTime	5
		(V15)	GROUPING: rentalNumber, rentalStartingTime, rentalBranchNumber, rentalRoomNumber, rentalDeposit	20 - 135
		(T2)	ORDERING: rentalNumber	10 - 100
41	RentalCustomer	(P10.1), (P21), (P28)	PREDICATE: rentalCustomerNum	10 - 130
		(P10.1), (P17), (P21), (P28), (V3), (V12), (V15)	JOIN: Rental **on** rentalNumber	30 - 285
		(P18), (P21), (V3)	JOIN: Customer **on** customerNum	30 - 210
		(P18)	PREDICATE: rentalCustomerRentalNumber	10 - 90
42	RentalPeriod	(P10.2), (V3)	PREDICATE: rentalPeriodStartingTime	20 - 155
		(P10.2), (P19), (V3)	PREDICATE: rentalPeriodEndingTime	20 - 165
		(P10.2), (P17), (P19), (V3), (V15)	JOIN: Rental **on** rentalNumber	30 - 350
		(P10.2)	ORDERING: rentalPeriodStartingTime	10 - 135
		(P19), (V3)	GROUPING: rentalPeriodEndingTime	10 - 30
44	RequestedRoomService	(P10), (V11)	JOIN: CustomerRoomService **on** serviceNumber	5 - 15

No.	Relations	Transaction	Field	Frequency (per day)
		(P11)	PREDICATE: requestedBillingNumber	30 - 200
		(P11)	PREDICATE: requestedLineNumber	30 - 200
		(P11), (V11)	PREDICATE: requestedServiceNumber	5 - 15
45	Reservation	(P7)	PREDICATE: reservationNumber	10 - 40
		(V6)	PREDICATE: expectedCheckInDate	5 - 50
		(V6)	PREDICATE: expectedCheckOutDate	5 - 50
46	ReservedRoom	(P20), (V6)	JOIN: Room **on** roomNumber **and** roomBranchNumber	10 - 115
		(V6), (V16), (P20)	JOIN: Reservation **on** reservationNumber	20 - 170
48	Room	(P4), (P20), (V6)	PREDICATE: roomBranchNumber	10 - 120
		(P4), (P20), (V6)	PREDICATE: roomNumber	10 - 120
		(P20)	ORDERING: roomNumber	10 - 60
		(V6), (V12)	JOIN: Branch **on** branchNumber	10 - 60
52	RoomUtilityRecord	(P6)	PREDICATE: roomUtilRoomNumber	10 - 135
		(P6)	PREDICATE: roomUtilBranchNumber	10 - 135
		(P6)	ORDERING: recordedTime	10 - 135
54	RoutineRoomService	(P13)	PREDICATE: routineServiceNumber	50 - 500
		(V11)	JOIN: CustomerRoomService **on** serviceNumber	10
		(V11)	GROUPING: routineServiceNumber	10
56	UtilityCost	(P6)	PREDICATE: utilityCostNumber	10 - 135
		(P6)	JOIN: CustomerBillingLine **on** customerBillingNumber **and** customerBillingLineNumber	10 - 135

Table 5 – Interaction between relations and query transactions

To conclude, we build the file organization and indexes for the database. Based on the interaction of data transactions and relations defined by frequencies above, **additional indexes are created as listed in table 6 below.** Please note **that primary key indexes** (e.g. unique clustered indexes) **are not considered** as additional indexes since, **according to Microsoft SQL documentation,** unique clustered indexes are automatically added by MS SQL DBMS; therefore, indexes listed in table 6 are entirely for non-key fields [2].

Created Indexes

No.	Relations	Index
4	Branch	branchName
		branchProvince
		branchNumber, branchName, branchEmail
9	Customer	customerNum, customerFirstName, customerMiddleName, customerLastName
		customerFirstName, customerMiddleName, customerLastName
		customerNationality
		customerDOB
10	CustomerBilling	billingNumber, billingDescription, billingCreatedTime
		billingCustomerNum
		billingFinancialTransactionNumber
		billingRentalNumber
11	CustomerBillingLine	customerBillingNumber, customerBillingTime
12	CustomerPassport	passportCustomerNum
		expirationDate
15	CustomerTelephone	customerTelephoneNumber
16	CustomerVisa	visaPassportNumber
		arrivalDate
17	Employee	employeeBranchNumber
		employeeFirstName, employeeMiddleName, employeeLastName
26	FinancialTransaction	financialTransactionAmount
27	InventoryEntry	entryReturningSupplyNumber
		entrySupplyNumber
		entryBranchNumber
		inventoryEntryType
35	PropertyDamage	propertyDamageMaintenanceNumber
		propertyDamageInspectionNumber
36	PropertyInspection	propertyInspectionRentalNumber
37	Purchasing	purchasingType
40	Rental	rentalStartingTime, rentalEndingTime
		rentalEndingTime
		rentalBranchNumber
		rentalBookingCustomerNum
		rentalRoomNumber
41	RentalCustomer	rentalCustomerRentalNumber
		rentalCustomerNum

No.	Relations	Index
42	RentalPeriod	rentalPeriodStartingTime
		rentalPeriodEndingTime
44	RequestedRoomService	requestedServiceNumber
		requestedBillingNumber
45	Reservation	expectedCheckInDate
		expectedCheckOutDate
46	ReservedRoom	reservedRoomReservationNumber, reservedRoomBranchNumber
52	RoomUtilityRecord	roomUtilBranchNumber
		roomUtilRoomNumber
		recordedTime
56	UtilityCost	utilityBillingNumber, utilityLineNumber

Table 6 – Additional Indexes added to certain relations in the Bright House database

There are several reasons why the indexes listed in table 6 were created. The first reason is that these indexed fields are frequently used by many transactions as listed in table 5. Another reason is that some of the fields are used together in aggregation, predicate, or grouping clause; therefore, indexes from multiple non-ordering key fields should be created [3]. Moreover, we also focus on indexing individual foreign key columns that are not primary keys since they are prominently used for searching in certain circumstances. For example, the **`InventoryEntry`** relation is usually used to search by **`entrySupplyNumber`** or **`entryReturningSupplyNumber`**; therefore, we create 2 indexes for each field.

User View Analysis

ORIGINAL DELIVERABLE REQUIREMENT

f. The results of the analysis of at least two user views for physical design (see page 582). The ER diagram in (b) may be used to specify the scope of the user views.

During the first phase of the database design project, three important user views are defined which include **`BranchManagement`**, **`BranchEmployee`**, and **`Customer`** user views. Each of the views is defined to have a different scope of the relation access. Two user views are as follows:

- **`BranchManagement`** consisting of Directors and Branch Managers of Bright House apartment.
- **`BranchEmployee`** consisting of employees in the position of accountant, cleaning personnel, and security.
- **`Customer`**, as the name suggests, consisting of customers who view branch and room information as well as making a reservation.

To further explain, the **`BranchManagement`** user view is specialized for directors and branch managers to get information that is crucial for managerial decisions. Moreover, this view is supposed to provide users with access to modify important or sensitive data such as branch, customers, rooms, customer payments, maintenance, payrolls, etc. On the other hands, the **`BranchEmployee`** user view is specialized for employees employed by branch managers; therefore, the data and transactions that are available will be related to operational data such as customer room services (for both routine and requested room service), property inspection, property damage, inventory. Moreover, since this view is designed to include accountants as well, financial transactions and purchasing are also made available. Lastly, the **`Customer`** user view is designed for the customer of the Bright House apartment to inspect the publicly available information of branches, rooms, reservations, and feedback.

Transactions by user views

In the following section, each transaction, as listed in the *Implementation List*, will be listed again to show which user views it belongs to. Some transactions may belong to multiple views since a transaction may be a combination of multiple requirements from different views. The fact that some requirements exist in multiple views can be another reason.

ID	Name	Description	Associated User Views
P1	spInsertNewBranch	Insert the details of a new Branch	**BranchManagement**
P2	spInsertEmployee	Insert the details of a new Member of an Employee	**BranchManagement**
P2.1	spInsertWorkingShift	Insert the employee working shift detail	**BranchManagement**
P3	spInsertCustomer	Insert the details of a New Customer	**BranchManagement**
P3.1	spInsertCustomerPassport	Insert the details of customers Passport	**BranchManagement**
P3.2	spInsertCustomerVisa	Insert the details of customers Visa	**BranchManagement**
P4	spInsertNewRoom	Insert the details of a new Room in a Branch	**BranchManagement**
P5	spInsertCustomerPaymentBillingLine	Insert the details of a new Customer Payment Billing line	**BranchManagement**
P6	spInsertUtilityCost	Insert the details of a new Room Utility	**BranchManagement**
P7	spInsertReservation	Enter the details of a new Reservation	**BranchManagement**
P8	spInsertRental	Enter the details of a new Rental	**BranchManagement**
P9	spInsertSubsequentMonthlyRentalPeriod	Enter Subsequent Monthly Rental Period	**BranchManagement**
P10	spInsertCustomerRequestedRoomService	Enter the data of a Requested Room Service	**BranchManagement**
P10.1	spGetRentalDuringTimeByCustomerNum	Get Current Rental by a Customer	

ID	Name	Description	Associated User Views
P10.2	spGetLatestCustomerBillingByRentalNum	Get Current CustomerBill by RentalNum	
P10.3	spGetLatestBillingLineByBillingNum	Get Latest BillingLine by BillingNumber	
P11	spInsertActionDoneInRequestedRoomService	Enter the data of an Action done within a session of Requested Room Service	**BranchEmployee**
P12	spInsertWithdrawalEntryRelatedToCustomerService	Add a Supply Withdrawal entry associated with Customer Service.	
P12.1	spIdentifyReturningSupplyOnHandQuantity	Retrieve the number of a returning supply so far given a branch.	**BranchEmployee**
P12.2	spFindBranchOfGivenEmployeeNum	Find Branch Number from a given employee number	
P12.3	spFindAccountantByBranchNum	Find accountant number by a branch number	
P13	spInsertRoutineRoomServiceAction	Enter the data of an Action done with a Routine Room Service	**BranchEmployee**
P14	spInsertNewSupplyFromPurchasingToInventory	Add a new Supply from a purchasing to the inventory	**BranchEmployee**
P15	spListBranchByRegion	List branches by a region	**BranchManagement**, **Customer**
P16	spListEmployeeByGivenDetails	List employee by given details	**BranchManagement**
P17	spListCustomerRentalsByTimeWindow	For each Customer, list their rentals that are within the given time window.	**BranchManagement**
P18	spIdentifyCustomersByRentalDetails	Identify Customers by Rental Details	**BranchManagement**
P19	spListExpiringRentalsInTimeRange	List expiring rentals in time range	**BranchManagement**

ID	Name	Description	Associated User Views
P20	spListFreeRoomInGivenTimeRange	List free rooms in a given time range	**BranchManagement**
P21	spListAllRentalsByCustomerNumber	List all rentals in any branches associated with a given customer	**BranchManagement**
P22	spListRoomServiceByRental	For a given rental, list all room service details including its category, total actions are done, starting time.	**BranchManagement**, **BranchEmployee**
P23	spInsertPropertyInspection	Insert the details of a Property Inspection	**BranchManagement**
P24	spInsertPropertyDamage	Insert the details of a Property Damage	**BranchManagement**
P25	spInsertMaintenanceTask	Insert the details of a Maintenance	**BranchManagement**
P26	spGetBranchEmployeeWithMaxWage	Get Branch employee with a max wage	
P27	spInsertRoutineRoomService	Enter the data of a Routine Room Service	**BranchEmployee**
P28	spListCustomerPaymentByRentalCustomer	List the customer payment by a given rental number and customer name.	**BranchManagement**
T1	trAddRoomUtilityOperand	Room Utility Operand Trigger	**BranchManagement**
T2	trAddRentalPeriod	Initial Rental Period Trigger	**BranchManagement**
V1	EmployeeDetailsView	List the details of each employee including gender, branch, wage, position, working hours.	**BranchManagement**, **BranchEmployee**
V2	EmployeePayrollView	Identify the total employee payroll for a day, week, or month.	**BranchManagement**
V3	CustomerView	Identify customer information as well	**BranchManagement**

ID	Name	Description	Associated User Views
		as his/her first rental branch.	
V4	SummarySupplyView	List the details of each type of supply for each branch	**BranchManagement**
V5	WeeklySupplySummaryView	Show the summary of supply weekly usage in each branch	**BranchManagement**
V6	RoomRentalStatusView	List every room in all branches as well as identify its occupation or reservation status	**BranchManagement, Customer**
V7	AvgCustomerDemographicPaymentView	List Average Payment is done by customers in each demographic (or nationality).	**BranchManagement**
V8	MonthlyHighestCustomerPaymentView	List Highest Customer Payment grouped by Month and Customer's nationality	**BranchManagement**
V9	MonthlyAvgCustomerPaymentView	List Average Customer Payment grouped by Month and Customer's nationality	**BranchManagement**
V10	MonthlyRentalFrequencyView	List the frequency of starting and the frequency ending rentals in each month	**BranchManagement**
V11	IndividualRoomServiceView	For each service instance, list all room service details including its category, total actions done, starting time.	**BranchManagement**
V12	BranchAvgRentalAndCustomerView	For each branch, retrieve information regarding rentals and customers such as rental duration, PAX, etc.	**BranchManagement**

ID	Name	Description	Associated User Views
V13	MaintenancePrecedingInspectionView	List all Maintenance instances and preceding inspections	**BranchEmployee**
V14	TopTenMonthlyBranchCustomerTransactionView	List Top-10 customer transactions in each month along with associated customer details	**BranchManagement**
V15	RentalSupplyView	List room number, customer name, rental type, deposit, rental amount, as well as supplies used by the room.	**BranchEmployee**
V16	CustomerReservationView	List the details of a Reservation that customers have made.	**Customer**
V17	BranchContactView	List the Contact details in each branch	**Customer**
V18	CustomerFeedbackView	List the Feedback that is submitted by the customer who gave the feedback	**Customer**

Table 7 – List of all transactions implemented along with related the user views that it belongs to.

Additional Note

Some of the transactions may not be associated with any user views since they are mainly called by other stored procedures.

Database Fine Tuning

In this project, on top of the SQL implementation, our team also focused on improving the performance of the database by looking at flaws in the implemented SQL commands. So far, we inspected all possibilities of fine-tuning in queries in terms of:

1. **Index**
   a. **Lack of indexes** – We have ensured the index implementation coverage by identifying the interaction of transactions and relations in table 5.
   b. **Unutilized Indexes** – Indexes were created based on fields listed in table 5 with its usage in mind. We also randomly inspected the execution plan of some of the transactions to check if the index is used.
   c. **Indexes on frequently modified attributes** – Most of the indexes are implemented on attributes that are not supposed to be modified due to the business rules. Some of the examples are `billingNumber`, `billingDescription`, `billingCreatedTime`.
2. **Design Tuning**
   a. We do not consider both **Vertical Partitioning and Horizontal Partitioning** in this project since it requires a considerable amount of workload in modifying schemas and data insertions. Moreover, it will also require extensive modification in the implemented transaction.
   b. **Attribute replication** is also ignored since our database already takes more than 336 MB on disk. Doing replication would mean the database would take more space and it means than we also have to implement more triggers (and other denormalization implications) to ensure data consistency.
3. **Query**
   a. `IN` clause is avoided in data retrieval queries to ensure the performance gain of the implemented indexes.
   b. Since most primary keys are integers, when using the JOIN clause, we can **avoid joining by using string equality comparison**.
   c. When checking for the existence of a record, `EXISTS()` is used instead of `IN` and `COUNT()` to ensure the performance gain [4].
   d. When retrieving data, data columns are explicitly selected instead of using wildcard selection (`SELECT * ...`) to ensure the reasonable uses of memory [4].

References

[1] T. Conolly and C. Begg, Eds., Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition, Harlow, Essex: Pearson Education Limited, 2015, p. 579.

[2] Microsoft Corporation, "Clustered and Nonclustered Indexes Described," Microsoft Corporation, 2 November 2019. [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-ver15. [Accessed 5 May 2020].

[3] A. M. Poolet, "Indexing Dos and Don'ts," ITProToday, 22 December 2002. [Online]. Available: https://www.itprotoday.com/sql-server/indexing-dos-and-don-ts. [Accessed 9 May 2020].

[4] K. Bloch, "SQL Database Performance Tuning for Developers," Toptal, 2013. [Online]. Available: https://www.toptal.com/sql-server/sql-database-tuning-for-developers. [Accessed 5 May 2020].

Submitted Script Files

In this phase, our team has submitted 7 SQL files which are:

1. **Improvements for Phase 3**
   All SQL commands in this script file need to be executed first. This SQL script file contains fixes in loaded data in phase 3 as well as some logical database implementations in terms of constraints.
2. **Script for Creating Indexes**
   All scripts related to **the creations of indexes** from table 6 are contained within this file. Executing commands in this file may take a while since some of the indexes are tied to relations that have many data tuples.
3. **Script for Creating Stored Procedures**
   - Each Stored Procedure is named *P1*, *P2*, *P3*, and so on.

   This SQL file contains the SQL commands to create ***Stored Procedures*** for data retrieval or relation modification such as inserting a new customer billing line.
4. **Script for Creating Triggers**
   - There are triggers named *T1* and *T2*, respectively.

   This SQL file contains the SQL commands to create ***Triggers*** for data modifications.
5. **Script for Creating Views**
   - Each View is named *V1*, *V2*, *V3*, and so on.

   This SQL file contains the SQL commands to create ***Views*** for data retrieval and data modifications.
6. **Script for Retrieving Data**
   This SQL file contains the ***example*** SQL commands to **retrieve data** from certain stored procedures and certain views.
7. **Script for Modifying/Updating Data**
   This SQL file contains the ***example*** SQL commands to **modify/update data** by using certain stored procedures and via certain views.